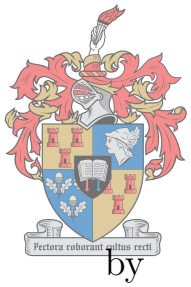
Development of a Big Data Analytics Demonstrator



by

Rhett Desmond Butler

Thesis presented in fulfilment of the requirements for the degree of Master of Engineering (Industrial Engineering) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. JF Bekker

December 2018

Declaration

By submitting this thesis electronically, I Rhett Desmond Butler declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2018

Acknowledgements

William Arthur Ward said. "*Feeling gratitude and not expressing it is like wrapping a present and not giving it*".

For this reason, I would like to express my gratitude and thanks to,

Annecke & Monet Butler,

my friends,

my love,

and finally, a big thank you to my project supervisor,

Professor James Bekker,

for your support and guidance through every stage of the project.

Abstract

The continued development of the information era has established the term 'Big Data' and large datasets are now easily created and stored. Now humanity begins to understand the value of data, and more importantly, that valuable insights are captured within data. To uncover and convert these insights into value, various mathematical and statistical techniques are combined with powerful computing capabilities to perform analytics. This process is described by the term 'data science'. Machine learning is part of data analytics and is based on some of the mathematical techniques available.

The ability of the industrial engineer to integrate systems and incorporate new technological developments benefiting business makes it inevitable that the industrial engineering domain will also be involved in data analytics. The aim of this study was to develop a demonstrator so that the industrial engineering domain can learn from it and have first-hand knowledge in order to better understand a Big Data Analytics system.

This study describes how the demonstrator as a system was developed, what practical obstacles were encountered as well as the techniques currently available to analyse large datasets for new insights. An architecture has been developed based on existing but somewhat limited literature and a hardware implementation has been done accordingly. For the purpose of this study, three computers were used: the first was configured as the master node and the other two as slave nodes. Software that coordinates and executes the analysis was identified and used to analyse various test datasets available in the public domain. The datasets are in different formats which require different machine

learning techniques. These include, among others, regression under supervised learning, and k-means under unsupervised learning.

The performance of this system is compared with a conventional analytics configuration, in which only one computer is used. The criteria used were 1) The time to analyse a dataset using a given technique and 2) the accuracy of the predictions made by the demonstrator and conventional system. The results were determined for several datasets, and it was found that smaller data sets were analysed faster by the conventional system, but it could not handle larger datasets. The demonstrator performed very well with larger datasets and all the machine learning techniques applied to it.

Opsomming

Die volgehoue ontwikkeling van die inligting-era het die term ‘Groot Data’ gevestig en reuse-datastelle word deesdae met gemak geskep en gestoor. Belangriker is dat die mensdom die waarde van data begin begryp, en meer nog, dat daar waardevolle geheime in data opgesluit kan lê. Om hierdie geheime te ontbloot en om te skakel sodat dit besigheidswaarde het word verskeie wiskundige en statistiese ontledingstegnieke tesame met kragtige rekenaarvermoë saamgespan vir ontledings. Hierdie aksie word beskryf deur die term ‘datawetenskap’. Masjienleer is deel van data-analitika en word baseer op sommige van die wiskundige tegnieke beskikbaar.

Die bedryfsingenieur se vermoë om stelsels te integreer en nuwe ontwikkelings tot voordeel van ondernemings in te span maak dit onafwendbaar dat die bedryfsingenieurswese-domein ook betrokke sal raak by data-analitika. Die doel van hierdie studie was om ’n demonstreerder te ontwikkel sodat die bedryfsingenieurswese-domein daaruit kan leer en eerstehandse kennis kan hê ten einde ’n Groot Data-stelsel beter te verstaan.

Hierdie studie beskryf hoe die demonstreerder as stelsel ontwikkel is, watter praktiese struikelblokke tegekom is asook die tegnieke tans beskikbaar om groot datastelle vir waarde te ontleed. ’n Argitektuur is ontwikkel gebaseer op bestaande, maar ietwat beperkte literatuur en ’n hardeware-implementering is daarvolgens gedoen. Vir die doel van die studie is drie rekenaars gebruik: een wat dien as die meester en twee as slawe. Programmatuur wat die analise kordineer en uitvoer is identifiseer en gebruik om verskeie toetsdatastelle wat in die openbare domein beskikbaar is, te ontleed. Die datastelle is in verskillende formate wat verskillende masjienleertegnieke vereis. Dit sluit in onder

andere regressie onder geleide leer, en k-gemiddeldes onder ongeleide leer.

Die prestasie van die stelsel is vergelyk met 'n konvensionele opstelling waarin slegs een rekenaar gebruik is. Die maatstawwe wat gebruik was, is 1) tyd om 'n datastel te ontleed met 'n gegewe tegniek en 2) die akkuraatheid van die voorspellings gemaak deur die demonstreerder en konvensionele stelsel. Die resultate is vir verskeie datastelle bepaal, en dit is gevind dat kleiner datastelle vinniger deur die konvensionele stelsel ontleed word, maar dat dit nie groot datastelle kan hanteer nie. Die demonstreerder het baie goed presteer met groot datastelle en al die masjienleertegnieke wat daarop toegepas is.

Contents

Nomenclature **xxii**

1 Introduction **1**
1.1 Background . . . 1
1.2 Rationale of research . . . 4
1.3 Problem statement . . . 4
1.4 Proposal . . . 5
1.4.1 Project objectives . . . 5
1.4.2 Research approach . . . 6
1.4.3 Deliverables envisaged . . . 6
1.4.4 Project sope . . . 6
1.4.5 Research strategy . . . 7
1.4.6 Data collection and analysis . . . 7
1.4.7 Research design . . . 7
1.4.8 Research methodology . . . 7
1.5 Thesis outline . . . 8
1.6 Proposition . . . 9

2 Literature study **10**
2.1 Big Data . . . 10
2.1.1 Big Data defined . . . 10
2.2 Big Data Architecture . . . 12
2.2.1 Big Data Reference Architecture methodology . . . 13
2.2.2 Big Data Architecture components . . . 15
2.2.2.1 Data extraction . . . 15

2.2.2.2 Stream processing . . . 17
2.2.2.3 Information extraction . . . 17
2.2.2.4 Manage data quality or uncertainty . . . 18
2.2.2.5 Data integration . . . 18
2.2.2.6 Data analysis . . . 19
2.2.2.7 Data distribution . . . 20
2.2.2.8 Data storage . . . 20
2.2.2.9 Metadata management . . . 21
2.2.2.10 Data life cycle management . . . 21
2.2.2.11 Privacy . . . 22
2.2.3 Lambda Architecture in Big Data . . . 22
2.2.4 Case studies: Big Data Architecture . . . 28
2.2.5 Summary of Big Data Architecture . . . 34
2.3 Big Data Analytics . . . 35
2.3.1 Analytics process . . . 37
2.3.2 Data Mining . . . 40
2.3.3 Machine learning . . . 43
2.3.3.1 Supervised learning: Regression . . . 46
2.3.3.2 Supervised learning: Classification . . . 60
2.3.3.3 Unsupervised learning: Clustering . . . 81
2.3.3.4 Reinforcement learning . . . 98
2.3.4 Dimensionality reduction . . . 99
2.4 Hadoop . . . 103
2.4.1 Method for storing data in Hadoop: HDFS . . . 105
2.4.2 Method for processing data in Hadoop: MapReduce . . . 107
2.5 Spark . . . 110
2.5.1 Method for processing data: Spark . . . 111
2.6 Semantic web and resource description framework in Big Data . . . 114
2.6.1 Ontologies . . . 114
2.6.2 Semantic web . . . 115
2.6.3 Resource description framework . . . 116
2.7 Benefits and drawbacks of Big Data . . . 117
2.8 Current features and trends in Big Data . . . 119

2.8.1 Current features of Big Data systems 119
2.8.2 Trends in Big Data 123
2.9 Synthesis of literature 124
2.10 Summary of literature 128

3 Proposed Big Data Architecture 130
3.1 Methodology used to develop the Big Data Architecture 131
3.2 Big Data Demonstrator Architecture 133
3.2.1 Stakeholders of the Demonstrator Architecture 133
3.2.2 Big Data Demonstrator Architecture goal 135
3.2.3 Big Data Architecture development 135
3.2.4 Proposed Architecture model and components 137
3.2.4.1 Data source 138
3.2.4.2 HDFS converting 138
3.2.4.3 Data analysing 139
3.2.4.4 Visualising the results 141
3.2.4.5 HDFS storage 141
3.2.4.6 Query result views 141
3.2.4.7 Analyst queries 142
3.3 Evaluation of the Proposed Architecture 142

4 System Design and Development 147
4.1 System hardware 148
4.1.1 Hardware selection 148
4.1.2 Coupling the computers to a network 150
4.2 System software 152
4.2.1 Operating system 153
4.2.2 Data storage solution 154
4.2.3 Data analytics solution 156
4.2.4 Programming language for BDA 158
4.2.5 Data visualisation solution 158
4.3 System Design conclusion 159

5 Big Data Analytics Demonstrator Experiments **161**
5.1 Big Data Analytics Demonstrator and Standard configuration . . 162
5.1.1 The standard analytics system 163
5.2 Algorithms used in the analysis of both systems 164
5.3 Datasets used to conduct the experiments on both systems 166
5.4 Analytics process used conducting the experiments 166
5.4.1 Results gathered from each of the experiments 168
5.5 The regression experiments 170
5.5.1 Data types required for regression 170
5.5.2 Logistic regression 171
5.5.3 Linear regression 173
5.6 The classification experiments 175
5.6.1 Data types required for classification 175
5.6.2 Decision trees 175
5.6.3 Naive Bayes 177
5.6.4 Linear support vector machines 179
5.6.5 Multilayer perceptron classifier 181
5.7 The clustering experiments 183
5.7.1 Data types required for clustering 183
5.7.2 k-means 184
5.8 Conclusion of the experimental results 189

6 Conclusion on the Big Data Analytics Demonstrator **193**
6.1 Summary of the project research and results 193
6.2 Future research in Big Data 196
6.3 Self-assessment of the research conducted 196

References **224**

A The process to configure Hadoop and Spark on a multinode cluster **225**
A.1 Connecting the computers together on the network 225
A.1.1 Securing the connection 228
A.2 Data storage solution 231

A.2.1 The Hadoop configuration files 238
A.2.2 The Hadoop dashboards 243
A.2.3 Working with the HDFS environment 247
A.3 Data analytics solution 247
A.3.1 The Spark dashboards 251
A.3.2 Creating the Spark environment 254

B Literature of the algorithms used by Spark and Scikit-learn 256
B.1 Regression machine learning techniques 256
B.1.1 Logistic regression algorithm in Spark 256
B.1.2 Logistic regression algorithm in Scikit-learn 257
B.1.3 Linear regression algorithm in Spark 258
B.1.4 Linear regression algorithm in Scikit-learn 258
B.2 Classification machine learning techniques 258
B.2.1 Decision tree algorithm in Spark 259
B.2.2 Decision tree algorithm in Scikit-learn 260
B.2.3 Naive Bayes algorithm in Spark and Scikit-learn 261
B.2.4 Linear support vector machines algorithm in Spark and Scikit-learn 262
B.2.5 Multilayer perceptron classifier algorithm in Spark and Scikit-learn 263
B.3 Clustering machine learning techniques 265
B.3.1 k-means algorithm in Spark and Scikit-learn 265

List of Figures

1.1 The challenge of determining from the amount of growing data is deciding what data holds value and what is 'noise'. 2

2.1 A Big Data reference architecture found in literature. 16
2.2 The three different layers that for part of the Lambda Architecture 23
2.3 The *recomputation* algorithm used in the Lambda Architecture . . 24
2.4 The *incremental* algorithm used in the Lambda Arhcitecture . . . 25
2.5 An architecture developed in order to provide a large-scale security monitoring system. 28
2.6 An arhitecture which provides a framework for the development of a Smart City Application. 30
2.7 The architecture on which the Amazon Web Services system is built. 31
2.8 An architecture to provide analytics to a smart-grid solution. . . . 32
2.9 An architecture aimed at analysing the large volumes of civil aircraft data being collected. 33
2.10 The different methods, phases and techniques involved in conducting Big Data Analytics 36
2.11 A visual representation of a data cube, with multiple dimensions and a value within each cell. 43
2.12 An example of a linear regression fitted to a dataset. 51
2.13 An example of a non-linear regression fitted to a dataset. 55
2.14 A decision tree which determines the likelihood of a person buying a computer given certain attributes. 65
2.15 A decision tree where pruning was employed in order to remove overfitting. 66

2.16 The structuring of a tree, with the objects in C. 67
2.17 A depiction of a neuron found in human brains, and how it translated into a ANN. 71
2.18 An example of a hyperplane that is created between two classes using the SVM technique. 73
2.19 The *car start problem* used to illustrate a causal network and show each causal link. 74
2.20 A simplified visual representation of a Bayesian Network. 75
2.21 The concept of *d-separation*. 76
2.22 A Bayesian Network example visualised. 77
2.23 An example of the result of clustering algorithm. 87
2.24 A visual representation of two points in a spatial data space. . . . 90
2.25 An example of an irregularly shaped cluster. 91
2.26 A visual representation of the rectangular grids created, wherein the data is contained. 92
2.27 Using grid cells, the STING algorithm collects statistical information on the two-dimensional data. 92
2.28 A visualisation of multiple computing nodes joined together to form a rack with data flows. 104
2.29 The HDFS data distribution and storage process. 107
2.30 The Map and Reduce process using the key-value pair notation. . 108
2.31 The Map and Reduce process executed within the *Hadoop MapReduce* environment. 109
2.32 The analysis process conducted when using RDDs in Spark. . . . 113
2.33 The Spark shell that is provided in Spark to conduct analysis, configured for Python. 114
2.34 A RDF triple consisting of a subject and an object made up of various properties. 117
2.35 The visualisation of the CAP theorem. 122

3.1 The different properties that are included to provide effective documentation of the architecture description. 132

3.2 The proposed Big Data architecture for the demonstrator, high level System Diagram 1 (SD1) view. 138
3.3 A zoomed-in view of the *HDFS Converting* process, shown is the System Diagram 2 (SD2) level view. 139
3.4 A zoomed-in view of the *Data Analysing* component of the projects architecture, shown is the System Diagram (SD2) level view. . . . 140

4.1 A visual representation of the hardware used in the projects Big Data Analytics Demonstrator. 149
4.2 The output given in the computer's network settings, indicating a connection to the university network. 151
4.3 A tree of the different software components of the Big Data Analysis Demonstrator. 152

5.1 The elbow plot of the cluster analysis on the BDAD for 195 clusters. 185
5.2 The elbow plot of the cluster analysis on the BDAD for five clusters. 186
5.3 The three clusters that formed from the k-means clustering on the BDAD. 187
5.4 The clusters that formed from the k-means clustering on the BDAD. 188

A.1 Running the *ifconfig* command to identify a computer's IP. 227
A.2 The results from running the *ping* command on a slave node, which show packets of information being sent and received. 229
A.3 The message provided to indicate a successful password-less login. 230
A.4 How the IP addresses and hostnames need to be added to the *etc/hosts* file in each node. 232
A.5 The output provided when checking the Hadoop version. 234
A.6 The output message indicating the NameNode is successfully formatted. 235
A.7 The output given when the Hadoop services are started. 237
A.8 The output given when the Hadoop YARN services are started. . 237
A.9 The configuration file *core-site.xml*, used in the Hadoop installation process. 238

A.10 The configuration file *hdfs-site.xml*, used in the Hadoop installation process. 239
A.11 The configuration file *mapred-site.xml*, used in the Hadoop installation process. 240
A.12 The configuration file *yarn-site.xml*, used in the Hadoop installation process. 241
A.13 The dashboard provided on the NameNode, indicating the Hadoop HDFS cluster capacity. 243
A.14 The dashboard provided on the third DataNode configured on the master node. 244
A.15 The dashboard provided on the first DataNode. 245
A.16 The dashboard provided on the second DataNode. 246
A.17 The output message when starting the Spark environment. 251
A.18 The first dashboard provided to a user with the status of the Spark configuration. 252
A.19 A dashboard provided by Spark to monitor individual spark queries. 253

List of Tables

2.1 The advantages and disadvatages of using *recomputation* and *incremental* algorithms . . . 26
2.2 Table comparing commonly used models for stream processing against different factors. . . . 27
2.3 Collection of software tools available for conducting data mining. . 44
2.4 Summary of regression techniques. . . . 48
2.5 A subset of data on the factors affecting the fertility of humans. . 58
2.6 Summary of classification techniques. . . . 62
2.7 A training dataset to demonstrate the ID3 algorithm. . . . 68
2.8 A collection of different applications for clustering. . . . 83
2.9 Summary of clustering techniques. . . . 84
2.10 An example of information used in movie recommender systems. . 94
2.11 The covariance matrix of the Iris dataset. . . . 101
2.12 The covariance matrix of the Iris dataset. . . . 102
2.13 The Principal Component values for the Iris dataset. . . . 103
2.14 The various Apache Hadoop projects, each designed for specific applications in the Big Data environment. . . . 106
2.15 The various Apache Hadoop projects. . . . 112
2.16 Table of the different semantic layers within the semantic web. . . 116

3.1 A table including the different stakeholders that are involved in this project and its architecture. . . . 134
3.2 Table of the different components used in the proposed architecture, and from which sources in literature and case studies these components were derived. . . . 143

5.1 The available Spark regression algorithms, along with the respective algorithms chosen to be used. 165
5.2 The available Spark classification algorithms, along with the respective algorithms chosen to be used. 165
5.3 The available Spark clustering algorithms, along with the respective algorithms chosen to be used. 165
5.4 A summary of the various datasets used to test the ML algorithms on both systems. 167
5.5 Results gathered from the analysis of the BDAD and standard systems with the notation used. 169
5.6 The results after conducting logistic regression. 171
5.7 The results after conducting linear regression. 174
5.8 The results after conducting a decision tree analysis. 176
5.9 The results after conducting a naive bayes analysis. 178
5.10 The results from conducting a SVM analysis. 180
5.11 The results after the multilayer perceptron classifier analysis. . . . 182
5.12 Summary of the experimental results from the different ML algorithms from the BDAD and SS. 190

Nomenclature

Superscripts

AD	Architecture Description
ANN	Artificial Neural Network
API	Application Program Interface
ARFF	Attribute-Relation File Format
ASF	Apache Software Foundation
AWS	Amazon Web Services
ZB	Zettabyte
BD	Big Data
BDA	Big Data Analytics
BDAD	Big Data Analytics Demonstrator
BI	Buisiness Intelligence
BL	Boltzmann Learning
BN	Bayesian Network
CAP	Partition Tolerance, Consistency and Availability
CART	Classification and Regression Trees

CL	Competitive Learning
CPU	Central Processing Unit
CRISP	Cross-Industry Standard Process for Data Mining
DAG	Directed Acyclic Graph
DIMM	Dual in-line Memory Module
DKNN	Dynamic k-Nearest Neighbour
DNS	Domain Name Servers
DRAM	Dynamic Random Access Memory
DT	Decision Tree
ECC	Error Correcting Code
ECL	Error-correction Learning
ECN	Explicit Congestion Notification
EMR	Elastic MapReduce
GB	Gigabyte
GFS	Google File System
GIS	Geographic Information Systems
GPU	Graphical Processing Unit
HDFS	Hadoop Distributed File System
HiveQL	Hive Query Language
HL	Hebbian Learning
HPC	High Performance Computing
HTML	Hypertext Markup Language

HTTP	Hypertext Transfer Protocol
I/O	Input/Output
Iaas	Information-as-a-Service
ID3	Interactive Dichotomiser
IG	Information Gain
IIG	Information Integration and Governanace
IP	Internet Protocol
ISO	International Organisation for Standardization
IT	Information Technology
JSON	JavaScript Object Notation
k-NN	k-Nearest Neighbour
KDD	Knowledge Discovery Database
KNNDW	k-Nearest Neighbour Ditance Weighted
LA	Lambda Architecture
LEM1	Learning from Examples Module version 1
MAE	Mean Absolute Error
MB	Megabytes
ML	Machine Learning
MPC	Mulitlayer Perceptron Classifier
MSE	Mean Square Error
NB	Naive Bayes
NN	Nearest Neighbour

NoSQL	Non-Structured Query Language
OCR	Optical Character Recognition
OPM	Object Process Methodology
OS	Operating System
Paas	Platform-as-a-Service
PCA	Principal Component Analysis
PIN	Personal Identification Number
RA	Reference Architecture
RAM	Random Access Memory
RDBMS	Relational Database Management System
RDD	Resilient Distributed Datasets
RDF	Resource Description Framework
ROD	Return on Data
RTT	Round Trip Time
Saas	Software-as-a-service
SEMMA	Sample, Explore, Modify, Model and Access
SNNB	Selective Neighbourhood Naive Bayes
SQL	Structured Query Language
SS	Standard System
SSD	Solid State Drive
SSE	Sum Square Error
SSH	Secure Shell

STING	STatistical INformation Grid-based
SU	Stellenbosch University
SVM	Support Vector Machines
TA-RMSE	Time Averaged Root Mean Square Error
TB	Terabyte
UC Berkley	University of California Berkley
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USD	United States Dollar
VDM	Value Difference Metric
W3C	World Wide Web Consortium
WAKNN	Weight adjusted k-Nearest Neighbour
WWW	World Wide Web
XML	Extensible Markup Language
XRFF	e-Xtensible-Relation File Format

Chapter 1

Introduction

1.1 Background

Currently, most humans own some form of electronic equipment, from cellphones to computers. Industries are attaching more sensors and installing systems to collect and remotely conduct performance tracking. All electronic devices send and receive data, which is stored on large data repositories typically in the control of large companies such as Google, Facebook, and Amazon. The devices allow for a more connected world, making it easier for a person to gain access to information about people, places and businesses. As stated by the founder of Facebook, Mark Zuckerberg, 'Facebook was not originally created to be a company. It was built to accomplish a social mission, to make the world more open and connected.' This connected world allows companies to analyse for example, trends and opinions of people and their interactions with products and/or services the company provides. Such information was previously not available for analysis. With the growing number of people using services such as Amazon, more data has become available to the company to analyse to improve their service to customers, as well as assisting in maintaining a competitive edge. This growing amount of data available to analyse, has helped shape what is now known as 'Big Data'. By capturing and analysing customer data, Amazon can, for example, determine the purchasing patterns of customers. This allows Amazon to in future focus on the items a customer will be the most likely to desire, (McAfee and Brynjolfsson, 2012).

To improve business output for any enterprise, low-cost computing and storage have allowed for larger-scale analytics to be performed on business processes, creating more accurate predictive models. Such analytics provide an enterprise with the ability to perform problem identification, future planning and performance tracking (Zikopoulos et al., 2012).

Figure 1.1: A typical challenge for enterprises with a growing amount of data becoming available is determining what data holds value and what data is considered 'noise' and needs to be discarded (Zikopoulos et al., 2012).

Due to the competitive nature of the private sector, the need to acquire and analyse larger amounts of data in order to maintain or create a competitive edge, has allowed for the growth in the data analytics sector and provided the need for 'Big Data'. Because of this trend in data analytics usage by various industries, Lohr (2012) states that there will not be a sector of business that in future will not be influenced by or use Big Data.

According to Zikopoulos et al. (2012), *volume*, *variety*, *veracity* and *velocity* are four of the characteristics that define Big Data. The *veracity* characteristic is not included in the definition given by Russom (2011) to define Big Data, but for this present study of Big Data it is included.

The *volume* characteristic refers to the amount of data being stored, which increases as a business or person collects more data. As stated by Zikopoulos et al. (2012), the amount of data collected and stored worldwide in 2009 was 0.8 ZB (zettabytes) and grew to 1 ZB in 2010. A zettabyte is equivalent to 1 billion

terabytes (TB) of data. *Variety* refers to the different data types captured; this includes structured data (where data is constrained to a given format, such as sensor data), semi-structured data and unstructured data (no given structure is imposed on the data such as text, video or audio data). The *velocity* component is the rate at which data arrives at an enterprise and is processed so as to draw meaningful conclusions from. Being able to immediately perform an analysis on data that arrived at an enterprise allows for a enterprise to maintain a competitive edge and have greater returns on data (ROD). ROD is a similar metric to return on investment (ROI) which is synonymous in the financial world with the returns an enterprise generates from investing in new pieces of machinery, for example. The *veracity* of data refers to the quality and trustworthiness of the data being collected. Data that typically cannot be trusted is seen as noise and needs to be removed for an enterprise to use only relevant data. The problem with this is to then first determine what data is 'noise' and needs to be discarded and for how long the raw or filtered data needs to be kept for.

Kambatla et al. (2014) identified factors which drive the collection and analysis of large amounts of data. The factors that were identified were systems complexity, efficiency improvements of enterprises and interactive or client-orientated systems. These, along with providing analytics which promote sustainability (*e.g.* identifying the low-cost point of maintaining healthcare infrastructure) have allowed for the growth that is being experienced in the field of data collection and analysis.

With the recent development in cloud computing, the structure of analytics (how analysis is performed) has shifted from infrastructure-as-a-service to data-as-a-service. This means Big Data Analytics are available at a lower cost (hardware) while the service provider continually improves the performance of the service.

Zikopoulos et al. (2012) make use of a gold-mining analogy to show the intrinsic value that big data holds. During the initial stages of mining, discovering gold is relatively easy and the costs are low. As the mining operation continues and less gold is visibly available, the more refining of the tons of dirt previously removed is required to extract finer particulates. This in turn requires more capital for machinery as well as time to process the tons of dirt. Similar to this,

the initial analysis of a large dataset requires less time and money to extract value from the data. To find deeper relationships between the data and possibly discover new results, further analysis of the 'waste' data is required. This does however cost more time and money, as well as more computing power and storage is required to retain the data that would otherwise be discarded.

1.2 Rationale of research

There are a growing number of research and implementations of Big Data projects each with different applications. These include using Big Data for autonomous or self-driving cars (large scale implementations thereof by Tesla and Google) and using Big Data to develop customer profiles when shopping online (examples are Takealot and Amazon) (Lohr, 2012). Big Data is also used by companies for traditional analytics applications. There is thus seen to be a growing market for Big Data projects.

Literature provides a wide range of tools, processes and methodologies by which to develop a Big Data system for a wide range of applications, from data mining to analytics (Zikopoulos et al., 2012). The research conducted in this project will provide the community with a demonstrator of such a Big Data system. The research conducted will provide a concise collection of current information of Big Data, along with a demonstrator of the capabilities thereof, which is lacking in the current literature.

1.3 Problem statement

There is a growing amount of data being stored due to storage costs decreasing and capacity increasing. In addition, companies are leveraging their Big Data for analysis thereby allowing for improved decision-making, as well as developing new technologies. The field of Big Data and its uses is thus ever-growing. Due to the relatively sudden uptake in Big Data, a demonstrator of a Big Data system for the industrial engineering community is lacking. This project makes use of current Big Data technologies and methods in order to develop such a demonstrator to

demonstrate the ability of Big Data to extract Business Intelligence (BI) from data.

1.4 Proposal

Herein is described the proposed problem definition, objectives, and scope of the project.

1.4.1 Project objectives

The following objectives are to be pursued in this project, namely:

I) To study literature relating to:

   a) previous implementations of Big Data projects,

   b) methods used in Big Data Analytics to generate results to a user,

   c) technologies to develop a Big Data Analytics system,

   d) trends in Big Data Analytics,

II) determine a suitable framework for the development of the Big Data Analytics tool. This information would be obtained from completing Objective I (a),

III) design and develop the Analytics tool which would aid a user in the decision-making process. Through using the information gathered in Objectives I (b) and (c) this objective would be completed.

IV) To verify and validate the Demonstrator developed in Objective III,

V) demonstrate the tool by using freely available data, to possible real-world applications the tool could provide,

VI) provide recommendations for future work that could be pursued after project completion and validation.

1.4.2 Research approach

The approach to be applied in this research project, using the research onion of Saunders et al. (2009) is categorised as *deductive*. The reason as to why a *deductive* approach is being followed is that the knowledge of the field 'Big Data' is researched, and using specific software programs, methods, tools and techniques a demonstrator is to be developed. Quantitative data is to then be used to test and validate the demonstrator, and thus a *mono-method* research approach is followed because only quantitative data is used.

1.4.3 Deliverables envisaged

The following deliverables are envisaged for this project:

- Thesis
- Article
- Big Data analytics program tool.

1.4.4 Project sope

A data analytics tool is to be developed which would serve as a Demonstrator of what Big Data is. The Demonstrator would take in structured datasets, which would then be pre-processed and analysed. The results of this analysis will then be compared to that of a traditional non-Big Data analytics system. The goal is to illustrate the benefits of Big Data Analytics, by showing the limitations of traditional analytical techniques. The analytics tool would make use of various functions or algorithms from machine learning, statistical, mathematical to predictive algorithms, in order to provide these insights in the datasets. The tool should be capable of processing data in batches/large datasets. Commonly used methods should be used to search and filter datasets, such that the tool would then be able to process the data.

1.4.5 Research strategy

Using the 'research onion' from Saunders et al. (2009), the research is to be conducted by firstly conducting a literature review, to gain understanding of what is currently included and understood under 'Big Data'. Case studies from literature will then be used to validate the Big Data Architecture, developed to guide the development of the Demonstrator. Publicly available data is then be used to test and validate the Demonstrator's ability to analyse a large datasets.

1.4.6 Data collection and analysis

For this project, *secondary* data is used to develop and validate the Big Data Analytics Demonstrator. This means the data has been transformed from the original in some way, however the project scope requires that the data should only be used to demonstrate the Big Data systems capabilities, and un-transformed data is not a necessity.

1.4.7 Research design

From Saunders et al. (2009), the *research design* for this project is categorised as *explanatory*. This is because existing knowledge and literature of the 'Big Data' field is researched and applied to a demonstrator which will use methods, tools and techniques to explain characteristics found in the data to gain new insights.

1.4.8 Research methodology

Here-in is described the proposed methodology in order to fulfil the objectives and reach the desired aim of the project.

To fulfil Objective I, a literature review will be conducted. The literature research will be carried out to better understand the origins of Big Data, the current state of Big Data within industry and how these systems are put together using an architecture. The literature will assist in focusing this project on a specific aspect of Big Data which will be pursued. The next step will be to determine from literature, the appropriate methods, algorithms and platforms in which to develop the analysis tool, fulfilling Objective II.

After the literature review stage of the project is complete, the next stage will be to begin development of the analysis tool in order to fulfil Objective III. First, a basic model will be developed using a small dataset (test data), to ensure the tool has the ability to provide an accurate analysis of the test data. After this, the tool will be expanded to accommodate larger datasets.

The next step in the successful completion of the project is to verify and validate the analysis tool. This is to ensure the tool developed is able to provide the user with meaningful results and that the results are correct. By doing so, Objective IV is fulfilled.

During the project, the system will be tested with large datasets in order to demonstrate the capabilities of the system compared to that of a non-Big Data system, fulfilling Objective V.

Finally, after Objectives III to V are completed, the final Objective VI is to be completed by the developer through reflection of the results so as to provide recommendations on improvements that could be made to the analysis tool and possible future work that can also be pursued, discovered during the project.

1.5 Thesis outline

- **Chapter 1:** This chapter serves as an introductory chapter to the project. The project background is discussed, thereafter the specific problem is identified, a project scope is outlined along with objectives wished to be achieved.
- **Chapter 2:** Literature surrounding Big Data, Big Data Architectures, Big Data Analytics, currently available Big Data technologies, trends in Big Data, benefits and drawbacks of Big Data, the semantic web, are included.
- **Chapter 3:** The Demonstrator architecture and methodology are outlined in this chapter, which will serve as a roadmap of how the Demonstrator is to be developed, and how the Demonstrator is to provide results.
- **Chapter 4:** The methodologies that are to be followed in order to develop the Big Data Analytics Demonstrator are outlined; this includes all system components and practical challenges that were experienced.

- **Chapter 5:** The Big Data Analytics Demonstrator is validated, the results are discussed, along with improvements to the results that can be made will be provided.
- **Chapter 6:** A conclusion as to the success of the project is given, what had been learnt and what future changes can be made to improve the demonstrator or build thereon.

1.6 Proposition

For this project, different aspects of Big Data are to be researched. The following is a short description of what is proposed. Firstly, to develop a Big Data analytics program using current methods and technologies to serve as a demonstrator to the Industrial Engineering community as to the applications of Big Data. Thereafter, provide a concise document of current research that has been done in the field of Big Data.

Chapter 2

Literature study

The concept of Big Data was introduced in Chapter 1. The proposed objectives, scope and deliverables envisaged were outlined and the research methodology for this thesis was developed. In this chapter, literature is researched so as to clearly define what is Big Data, the benefits and drawbacks of Big Data and how Big Data fits into the simulation and industrial engineering context. Also researched are the current features and future trends in Big Data and Big Data Analytics, previous implementations of Big Data Analytic systems and finally, the tools, architecture required, software and methods used in developing a Big Data Analytics tool. The tools and methods researched will ultimately be used in the Demonstrator developed in this thesis.

2.1 Big Data

To gain a better understanding what is big data and how it is defined, this section focuses on the four Vs found in literature. This will provide context and highlight some of the benefits and drawbacks associated with the use of a Big Data system.

2.1.1 Big Data defined

The four Vs: *Volume*, *variety*, *veracity* and *velocity* are common terms used widely in literature, each with variations to describe the components of Big Data.

- Volume: The amount of data currently moving across the internet every second, is larger than the entire volume of data stored on the internet 20 years ago, and the amount stored is said to double every 40 months as stated by McAfee and Brynjolfsson (2012). The 'Big' in Big Data originated because of the increasing amount of data being stored and available for analysis.

- Veracity: This term refers to the trustworthiness and quality of data, and forms part of the Big Data definition. With spam-bots and directed tweets on twitter, certain data is of value while other data collected and stored might not hold any value and needs to be discarded, (Zikopoulos et al., 2012).

- Variety: Data which is generated and collected from sources such as Facebook or Twitter are classified as *unstructured* data, this is because such data typically does not conform to a given structure. *Structured* data however, as found in a sensor located in machinery, or log data generated from executing an action such as purchasing an item on Amazon does conform to a specified structure. The log data typically contains the transactional information, (Zikopoulos et al., 2012). Data can come in three types, *structured*, *semi-structured* and *unstructured*. As examples of *structured* and *unstructured* data were provided, examples of semi-structured data include XML and JSON documents or NoSQL databases.

- Velocity: Data can be described by how it is delivered. Data can arrive either in batches which can then be analysed, or in streams (near-real time) for analysis during the arrival of the data, (Russom, 2011).

Some definitions add more terms to help describe Big Data in further detail. Katal et al. (2013) use the above terms along with the following to define big data:

- Complexity: This makes reference to the complicated process required in Big Data to filter, link, match and transform data which arrives from different sources. After this, data needs to be sorted, hierarchies created,

and relationships and correlations made, which can quickly become very complicated.

- Value: Through analysing and running queries, meaningful results can be discovered which in turn add value to a business.
- Variability: Inconsistencies exist within data flows, especially unstructured data such as social media data which cause peak data loads when certain events occur (*e.g.* when a sporting event is being held).

2.2 Big Data Architecture

The architecture to develop a Big Data analytic system is researched, as it will allow for the identification of various steps required in order to realise such a system. First, there is research into what a software architecture is, thereafter the requirements of a Big Data architecture through reference architectures, and finally a brief overview of previous Big Data architecture implementations.

Taylor et al. (2010) defines software architecture to be a set of principal design decisions made about a system. Further, in defining a software architecture, three building blocks are identified which constitute an architecture namely, *processing elements*, *data elements* and *connecting elements*. Taylor et al. (2010) describes how software architectures are captured using a model. A model in turn describes the design decisions made and fulfils all or certain functions of the software architecture.

Building on this definition of an architecture by Taylor et al. (2010), a similar definition is made by Maier (2013). In this definition, an architecture and more specifically a 'reference architecture' is an abstract term that describes a system which is designed to solve a specific problem given a specific context. Such an architecture is made up of 'system components', and describes how they interact with one another, given properties and functionalities that are provided from a set of business goal(s) and stakeholder requirements.

The difference between an 'architecture' and a 'reference architecture' is that the latter can be used in future developments of system architectures and is an abstract platform from which to develop an architecture (hence the use of

reference). An 'architecture' is a specific set of structures which together fulfil a specific objective(s) to achieve a specific goal.

The stakeholders of any software architecture are also deemed important, the *software architect* is the principal designer and models, assesses and evolves the architecture. Taylor et al. (2010) further states that a significant determinant of a project's ability to fulfil its objectives are whether there was an effective systems architecture. This is where *software managers* as stakeholders, need to provide project oversight and *software developers* who are also seen as stakeholders, need to realise the principal design decisions of the architecture.

2.2.1 Big Data Reference Architecture methodology

The development of any architecture involves a number of necessary steps in order for it to be realised. Galster and Avgeriou (2011) developed six steps which can be followed to ensure a successful development of a reference architecture and which are also used by Maier (2013) in the development of a 'Big Data Reference Architecture'. This discussion of the methodology is an expansion upon the work of Butler and Bekker (2017), which used the findings of Galster and Avgeriou (2011) and Maier (2013) in the development of an architecture. The steps are as follows:

1. Establish the reference architecture type
2. Select the design strategy
3. Conduct Empirical acquisition of data
4. Development of the reference architecture
5. Enabling a reference architecture with variability
6. Performing an evaluation of the reference architecture

The first step [1] is to ensure the architecture is aligned to the desired goals set out by stakeholders of a given domain. This includes determining the intended application and that high-level design specifications be established. In Maier (2013) a diagram is used to show this relationship.

The second step [2] involves establishing if the architecture is to be developed anew, or based on existing architectures in the domain. From this follows the collection of empirical data in step [3], which is necessary to start the development and testing in step [4] of the architecture. During this step, the architecture is required to be designed such that when applied it allows for a certain flexibility and variability as required in step [5] to ensure it does not have limited scope and focus. Finally, the evaluation step [6] is to validate whether or not the architecture can indeed be used.

Given that Maier (2013) focused on the development of a Big Data reference architecture, further consideration is given to his results and conclusions.

After establishing the methodology, the requirements for the architecture were determined. Using the familiar Vs definition of Big Data, Maier (2013) aligned the goals of the architecture around these Vs. The goals are then translated in requirements where for each a detailed requirement overview is given. The 'Vs' are as outlined in section 2.1.1, and were analysed according to a set of attributes. Those from Maier (2013) are:

- Requirement ID: Identifies the requirement.
- Requirement Type: Describing whether the requirement is *functional* or *Non-functional*. A *functional requirement* describes a functionality delivered to an end-user at the application level. A *Non-functional* requirement is functionalities not delivered to an end-user but essential for the operation.
- Goals: The high-level goals the requirement is supporting.
- Parent Requirement: The parent requirement in the requirement hierarchy.
- Description: Short summation of the requirement.
- Rationale: Justification given for the requirement.
- Dependencies: Describing any dependencies on other requirements.
- Conflicts: Conflicts between other requirements
- Literature: Literature that can support the requirement.

From these requirements and attributes outlined, Maier (2013) developed the following high-level Big Data reference architecture functional view. Following on from this, is a implementation oriented view, which is not included here but can be found in the article by Maier (2013).

2.2.2 Big Data Architecture components

The components that make up the reference architecture outlined in section 2.2.1 by Maier (2013) are now discussed further. The components are also outlined in Agrawal et al. (2012). The functional components of the reference architecture are as from Figure 2.1:

- Data Extraction
- Stream Processing
- Information Extraction
- Manage data quality/uncertainty
- Data Integration
- Data Analysis
- Data Distribution
- Data Storage
- Metadata Management
- Data Lifecycle Management
- Privacy

2.2.2.1 Data extraction

In order for data to be processed later by an analyst, the first step involves data being extracted from different sources, by making use various extraction methods. Data that was gathered does not need to conform to a specific data type and can be *structured*, *semi-structured* and *unstructured*. During these processes, methods to sort and filter the data are also conducted. Witten et al. (2016) make use of *data warehouses* which provide a single point of access to the extracted data, but they note that not all data useful for analysis is stored on warehouses and will depend on the needs, if data were required to be extracted from other sources, web pages, data stores *etc.* Further, it was stated that the degree of aggregation is an important factor during the data extraction and collection process. Using the example of Witten et al. (2016), a telecommunications company desires to

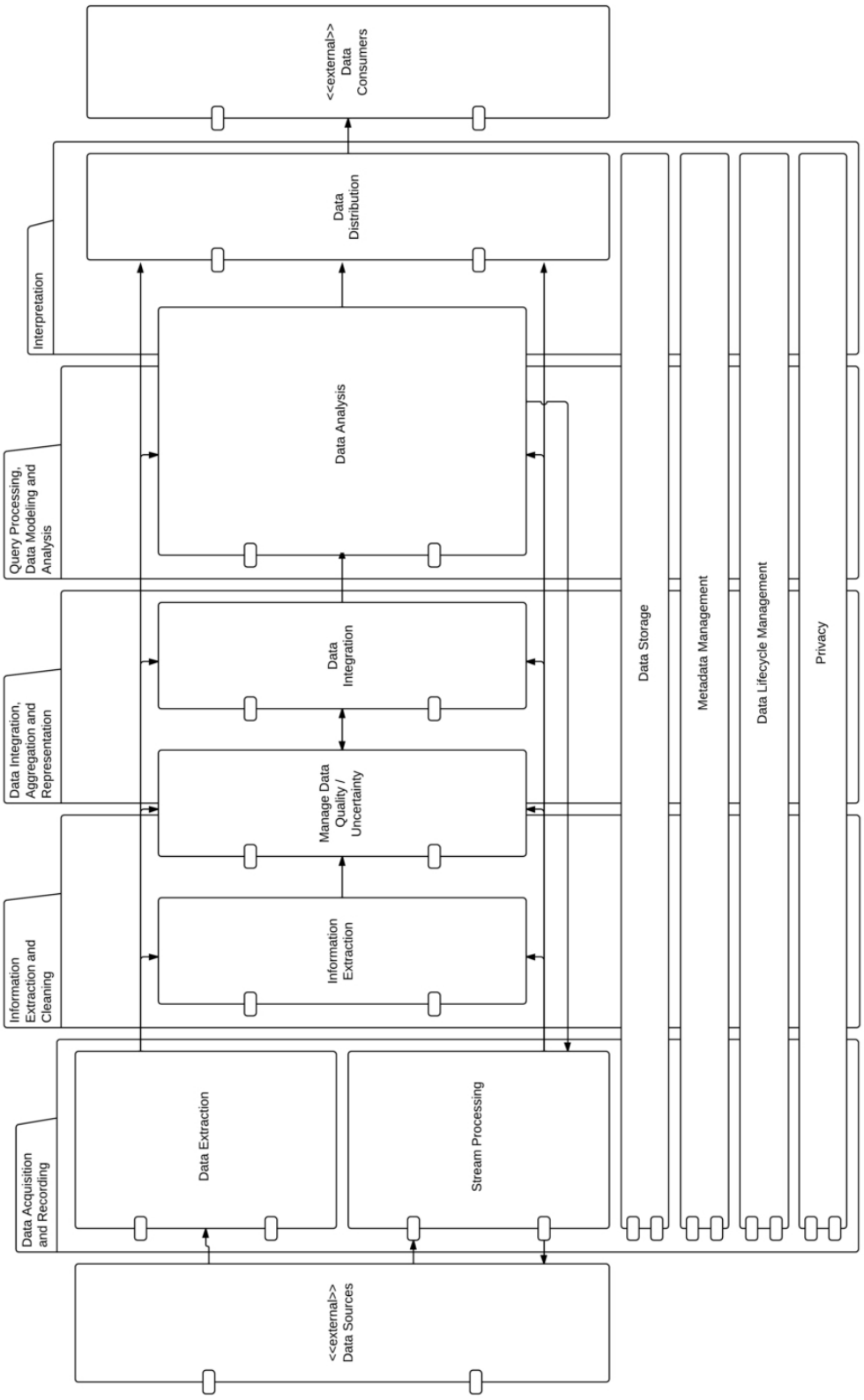


Figure 2.1: A Big Data reference architecture developed by Maier (2013) which can be used to develop a Big Data Analytics system.

study client behaviour, therefore raw call log data needs to be aggregated to a monthly or quarterly basis, and for a number of time periods in arrears.

The methods by which to extract data vary, as data can be extracted from various sources: file-based interfaces, standard or proprietary database interfaces or by extracting data from relevant web pages. An example of an input file format used is the ARFF (Attribute-relation file format) which accommodates nominal and numeric data types; or XRFF which provides the information and headers in an XML (eXtensible Markup Language) format (Witten et al., 2016).

2.2.2.2 Stream processing

To collect and analyse data from different sources in almost real-time, a process has to be defined. The process is defined by Maier (2013) to include two components, the first is data stream acquisition and the second data stream analysis. Before the final batch analysis (at rest analysis), the *acquisition* comprises of collecting the data from sources relevant to the analysis, thereafter conducting storing and filtering of the data. Finally, the data is stored or removed. The data is typically temporarily stored for a given time period (user dependent) after the analysis, to be used in future.

The *stream analysis* differentiates itself from batch analysis by analysing data in near real-time, in a fully automated manner. This is due to continually new data being added that needs to be processed, as characterised by stream processing. Datasets arriving for analysis continually grow and vary in type, Maier (2013) continues that for this reason, approximation techniques are required as well as continued feedback of results. This then allows for prediction and analysis improvement. Aggarwal and Wang (2007), Babcock et al. (2002) and Bonino and De Russis (2012) provide methods and algorithms by which to analyse real-time data. Some of the tools and techniques used therein are discussed further in section 2.3.3.

2.2.2.3 Information extraction

Regardless when working with different data types other than *structured* such as *semi-structured* (*e.g.* XML and HTML) and *unstructured data* (*e.g.* MP4, MP3),

the data containing meaningful information needs to be separated.

The process in order to do so, places a structure on the data by performing classification, clustering, entity recognition and relation extraction (Balke, 2012). These techniques by which to analyse and extract *unstructured* data are also presented in section 2.3.3.

2.2.2.4 Manage data quality or uncertainty

Functions fulfilled by this component include data cleaning, data correction, data completion and identification, and the removal of errors and noise in the datasets. This process addresses data quality problems of single data sets, whereas the *Data Integration* component makes use of techniques to reconcile data from different data sources.

To allow for the data quality management process, statistical and machine learning (ML) techniques are employed. These techniques allow for the cleaning of data up to a given point, but are specific to the data source. A brief overview of different data quality management objectives is provided, derived from Maier (2013).

First, *value completion* involves statistical and machine learning techniques to fill incomplete values. *Duplicate filtering* is conducted to remove duplicate data points that refer to one object or entity. Filtering does overlap with data integration as stated, both to identify and resolve identical objects or entities. *Outlier detection and Smoothing* is used to identify outliers to datasets and errors, then a correcting or smoothing is undergone on these errors. To overcome such challenges, it is suggested to make use of machine learning and statistical rule-based techniques within this component. *Inconsistency Correction* ensures referential integrity, which is important when attributes are given constraint(s) and the constraints must not be violated.

2.2.2.5 Data integration

Similar to data warehousing, which provides a repository separate to that of an organisations operational databases, data integration integrates multiple data sources together to be queried at once (Jiawei et al., 2012) in an overarching

schema. Schema integration, which consolidates multiple datasets into a global dataset, can be achieved through fixed mapping and transformation rules (Rule-based) techniques or by using semantic web and ontology techniques. These techniques in turn describe the different datasets' relation to each other.

When working with the lower level integration of data, 'entity resolution' can be applied for joining or reconciling data that refer to the same entities. Approaches for entity matching are discussed further by Fan et al. (2009) and Ngomo et al. (2013). If there are instances where entities that share attributes have opposite values, 'data fusion' techniques can be applied, as discussed by Bleiholder and Naumann (2008) and Srivastava and Dong (2013).

2.2.2.6 Data analysis

Data analysis is the component of the reference architecture which takes the extracted and integrated data, and then applies analytical techniques to derive meaning from the data. Data analysis, according to Maier (2013) can be conducted by the end user to provide insight into the data, or be used during data preparation, or calculation of results and storage.

By separating *Value deductions* and *Deep analytics* tasks from the presentation computations, high-performance computations (largely batch processing/-computations) can be performed. The end user-facing analytics such as *reporting and dash-boarding* solutions provide a user with overall status reports, while *free ad hoc analysis*, allows a user to freely query data and conduct their own computations.

Value Deductions in analytics involves adding information with a single or small record as input. This is characterised by adding key figures to the data, or enriching the master data. This is seen as data transformation and transferral of the data to data stores, used by end user-facing analysis tools.

Using machine learning and statistical tools and techniques, *Deep Analytics* can be conducted on large datasets, discussed in section 2.3.3, applying *batch analysis*. The results, insights, rules and models developed during deep analytics tasks are stored for use by end user-analysis functions.

Reporting and Dash-boarding is the component used after analysis, through which the use of visualisation methods, results and insights gained are shared. Such functionality is typically provided to executives, to give an overview of the data analysis that was conducted.

Free Ad hoc Analysis allows a user to randomly query data as desired. This increases the ease at which an analyst can sort through data to rapidly extract the results desired. The user is however required to have as stated by Maier (2013) to have the necessary skills and knowledge in machine learning, statistics and declarative languages (SQL, Python, Java *etc.*).

Data Discovery and Search is less focused on analysing data and more on filtering through data stores and sources. The ability to do so, allows the user to decide the appropriate analysis.

2.2.2.7 Data distribution

Functionalities this component provides include a *user interface* and *application interface* which supply results graphically or through reports, as required. Both of these output the results of the analytics process on a separate application and interface from which to derive meaning and make decisions.

2.2.2.8 Data storage

Data storage is stated by Maier (2013) to be a supporting function having no influence on the analysis, as data storage is undergone on input data and the results. Data storage includes all states of storage, from temporary to caching. Sub-functions include *staging*, *data management orientated storage*, *sandboxing* and *application optimised storage*.

Staging is to temporarily store data after it is extracted, for cleaning and filtering purposes, this allows for faster access to raw data (computation is de-coupled from source). After staging the data, the data is moved to a data store or warehouse. *Data Management orientated storage* refers to long-term storage, where data is cleaned and integrated, otherwise known as a data warehouse. *Sandboxing* is a temporary data store which is created for a user to experiment with data. The data to be experimented on is copied onto sandbox data stores

which are separated from the data warehouse, preventing any negative impacts on data stores during testing and experimentation.

Application Optimised Storage is storage allocated for the use by different analysis objectives and applications. The data stored herein is merely a subset of all data required by an application, whereby the data is enriched and transformed using deep analytics.

2.2.2.9 Metadata management

Metadata is simply data which describes data, and this component includes the extraction, storage, creation and management of structural, process and operational metadata, according to Maier (2013). Further, metadata management is a vertical function (high level function with different actions executed below each) *metadata extraction*, *metadata storage*, *provenance tracking* and *metadata access*.

Metadata Extraction is similar to data extraction, but is focused on the extraction of metadata from various sources in HTML, XML formats and RDF files *etc.* and is discussed further in Miller and Michael (2013) and Agrawal et al. (2012).

Metadata storage is simply providing a repository dedicated to storing all metadata.

Provenance tracking is the extraction and collection of metadata pertaining to operational and administrative data generated during the various data processing phases (job logging, component run times, volume of data and timeliness of loaded data). Through statistical and machine learning techniques, this data is used to determine the reliability and relevance of data sources and their results, also discussed in Agrawal et al. (2012).

Finally, *metadata access* is allowing administrators to access, manipulate and enhance the stored data (adding definitions, terms *etc.*).

2.2.2.10 Data life cycle management

This component encompasses the activities related to the management of data across its life cycle (creation to discarding). The overarching component that

forms part of data life cycle management is *Rule-based data and Policy tracking*. Life cycle management tasks are automated through the use of rule-based techniques, such as data archiving, compression and discarding. This means also moving data from an in-memory database to long-term disk-memory, (Maier, 2013).

2.2.2.11 Privacy

This component is included as it ensures the security and privacy of the data collected by means of *authentication*, *authorisation*, *access tracking* and data *anonymisation* methods. Authentication and authorisation require users to provide identification traditionally through user names and passwords assigned to each 'authorised' user and can limit the user's ability to search and extract data.

Access tracking is used in conjunction with authorisation and authentication, where the users who have logged in have their requests or log files tracked. This is used to ensure the user only accesses the data for which permission has been granted. Anonymisation is used to protect users by manipulating data fields, changing values, aggregation *etc.* before the data and results are presented to them. Zikopoulos et al. (2012) discusses the *Information integration and governance (IIG)* business strategy on how data should be treated in order to ensure its security and privacy.

There are clearly a wide variety of components that constitute a Big Data (Reference) Architecture as identified by Maier (2013) and which need to be considered when developing an architecture for a given application. Each of these components shown in Figure 2.1 is included in order to ensure the success of the Big Data System.

Next, a discussion follows on the *Lambda Architecture* which is a commonly used architecture for Big Data projects (Kamdar et al., 2014). Lastly, case studies are provided of Big Data Architectures developed for different applications.

2.2.3 Lambda Architecture in Big Data

Marz and Warren (2015) proposed the architecture to best answer queries an analyst would have when using a data analytics system. The architecture is

therefore designed to address the properties of various queries as well as ensuring the system to which these queries are submitted, is fault tolerant towards user input. The properties of a query that need to be addressed are, the *Latency* (how long it takes to run a query) of a query, the *Timeliness* (how up-to-date queries are) and finally the *Accuracy* of the query (how accurate does the query result reflect reality). To best address these concerns along with being fault tolerant, Marz and Warren (2015) proposed the *Lambda* architecture containing three *layers*.

The three architectural 'layers' as seen in Figure 2.2 are, the *Batch Layer*, *Serving Layer* and the *Speed Layer* (Rusitschka and Ramirez, 2014), (Marz and Warren, 2015) and (Kiran et al., 2015).

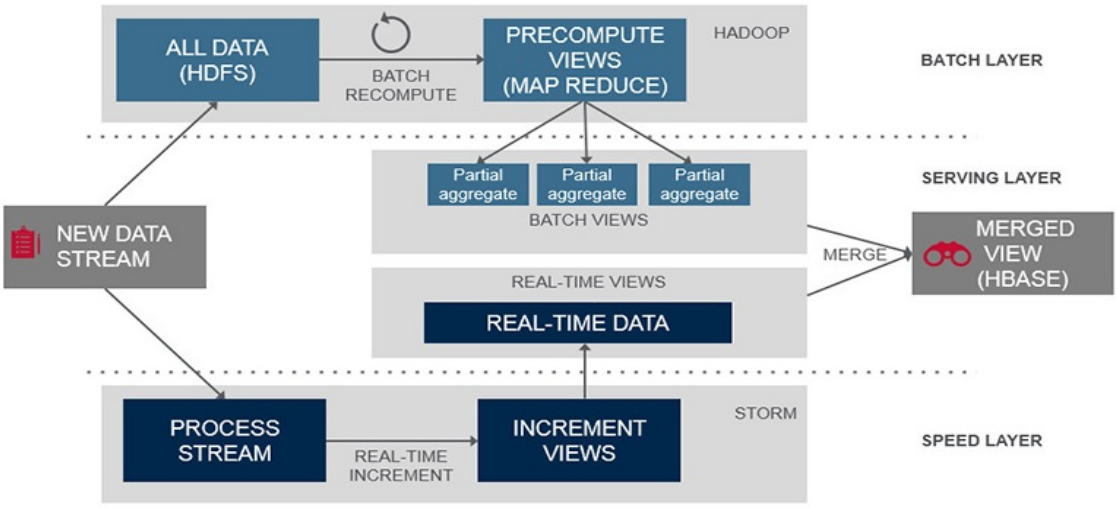


Figure 2.2: The three different layers that form part of the Lambda Architecture, illustrating how data is processed and analysed (MapR Technologies, 2017).

The *Batch layer* computes all data collected and stores the results as a set of 'views' on which the results are stored. Queries are then run by the analyst Liu et al. (2014) on these 'views'. Using the terminology from Marz and Warren (2015), a *batch-view* is a view of all the data after conducting processing on a given dataset. It is therefore a function of all the data and a query is a request made on the *batch-views*. The methods used to conduct the analysis of the 'batch' or data at rest are discussed further in section 2.3, this includes the processes

executed to the analytical techniques employed. The *Speed layer* makes use of parallel processing and conducts continuous processing of near real-time data and outputs the results, similar to the *Batch layer* as 'views'. These views are then queried by an analyst (Rusitschka and Ramirez, 2014) and (Liu et al., 2014). The *serving layer* is designed such that any query against the data made by an analyst is answered in this layer. This layer contains pre-computed 'views' and has low-latency, allowing fast replies to queries (Kamdar et al., 2014).

In the *Batch layer* the master dataset is precomputed into a batch of views to answer queries with low latency. All recomputations and current computations make use of *Big Data Analytics* to create the views for each layer. The analytics to develop the 'views' can use either *Recomputation* or *Incremental* algorithms, according to Marz and Warren (2015).

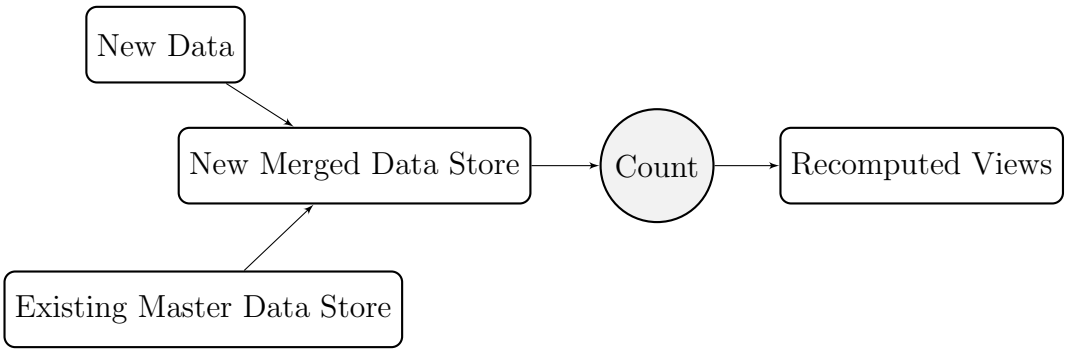


Figure 2.3: The *recomputation* algorithm which ingests new data, then conducts a recomputation of the entire data store, giving a new updated 'view' (Marz and Warren, 2015).

Recomputation algorithms shown in Figure 2.3 employ a strategy of removing old 'views' (results) and re-calculating the views for the entire dataset (after new data have been added). The *incremental* algorithms shown in Figure 2.4 update the views of the new data being added, these views are then added to the old views that have already been stored (cached). Marz and Warren (2015) continues by stating that to ensure human-fault tolerance, *recomputation* algorithms need to be created.

A computing paradigm used in the *batch layer* which allows for fault-tolerant and scalable computations is *MapReduce*, discussed in further detail in section

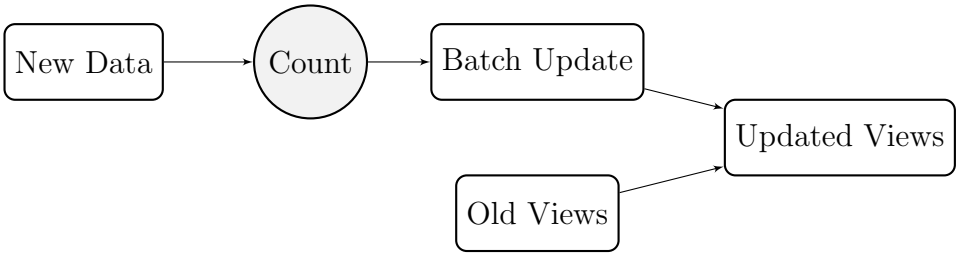


Figure 2.4: The *incremental* algorithm generates new views and adds these to the existing views store (Marz and Warren, 2015).

2.4.2. It can be employed along with any of these computation techniques (algorithms).

For storage in the *batch layer*, Marz and Warren (2015) discusses the two common methods, using a *key/value* store and the *distributed file system*. *Key/value* storage systems make use of a unique identifier, the key, which is linked to a piece of data (value). Relational databases are an example of such a system. Marz and Warren (2015) however state that such a storage system makes the data *mutable* (original data changed) as well as increasing storage costs (indexing required) and reducing processing performance (due to the reading and writing to-and-from the master dataset).

The *distributed file-system* according to Marz and Warren (2015) is a more efficient method of storing the data, allowing the data to be stored sequentially in blocks across multiple discs. This allows the data store to be human-fault tolerant, an important aspect of the *batch layer*. The *distributed file-system* allows for permissions to be set on the datasets, providing *immutability* (no changes made to the dataset) on the master dataset, which ensures that when conducting the analysis it is done on the original data and new data. Additionally, the original data is not 'corrupted' by the newly added data which might skew results.

A widely implemented *distributed file system* is shown in Figure 2.2. The *Hadoop HDFS* is available to be used in the architecture as it allows for the data to be stored so that it is scalable, fault-tolerant and able to make use of *MapReduce* parallel computations on commodity hardware Marz and Warren (2015). It is also available as open-source code (free to use) (Kiran et al., 2015). A detailed discussion of the *Hadoop HDFS* follows in section 2.4.1 where research

shows how the *HFDS* stores data so as to provide a scalable and fault-tolerant data storage solution.

Table 2.1 lists the advantages and disadvantages of employing a recomputation or incremental algorithm strategy in the *batch layer*.

Table 2.1: The advantages and disadvantages of using *recomputation* and *incremental* algorithms from Marz and Warren (2015).

Algorithm	**Advantages**	**Disadvantages**
Recomputation	Tolerant of human errors due to views being rebuilt, algorithm complexity results in simple batch views and low-latency.	Computationally intensive.
Incremental	Increases efficiency of the system, provides near real-time analysis and results and is less computationally intensive.	Requires special tailoring of algorithms and makes real-time algorithms complex. Estimations are required to be made of the data due to near real-time nature of analysis.

The goal of the *Speed layer* is to process data streams and update the 'views' and therefore makes use of the *incremental* algorithm strategy because of the rapid nature associated with analysing incoming data streams (Marz and Warren, 2015). The two models of stream processing are *one-at-a-time* (data is analysed at a first come fist serve basis) and *micro-batched* (analysing datasets divided into smaller datasets). Table 2.2 is a comparison table between *one-at-a-time* and *micro-batched* developed by Marz and Warren (2015). The requirements of the system that need to be taken into account and factors listed in Table 2.2 of the system, can then be used to guide the appropriate model(s) to be chosen.

Due to the scope of the project which is focused on analysing data at rest using *batch processing*, further research into *stream processing* methods is not carried out, but can be found in Marz and Warren (2015), for further reading.

In the Marz and Warren (2015) discussion of *stream processing*, the *storm* processing system is referenced as the preferred system to conduct real-time processing. Jones (2012) further discusses the key differences between *storm* and

Table 2.2: Table comparing commonly used models for stream processing against different factors, from Marz and Warren (2015).

Factors	***One-at-a-time***	***Micro-batched***
Low latency	✓	
At-least-once semantics	✓	✓
Exactly-once-semantics		✓
Simpler programming model	✓	
High throughput		✓

traditional databases such as *HDFS*, and how the *storm* model conducts processing. A use case of the *storm* processing model is Twitter (Jones, 2012), which uses *storm* to analyse tweets in near real time.

The *serving layer* seen in Figure 2.2 is designed to provide an interface for pre-computed and indexed 'views', generated in the *batch layer* and *speed layer* (Kamdar et al., 2014). An analyst can then submit queries to the *serving layer* which are processed with low latency due to being pre-computated, and offers high throughput. The requirements for the *serving layer* according to Marz and Warren (2015) are:

- Batch writeable: Newly made batch views are created, swapping out the old views for these new views (results).
- Scalable: The views need to grow and shrink to any size (amount of results stored should not be limited) while storing the data on multiple machines.
- Random reads: Allow a user to run queries and gain access to any part of the views.
- Fault-tolerant: Due to the distributed nature, machine failure needs to be accounted for.

Marz and Warren (2015) continue by suggesting open-source database solutions such as 'ElephantDB', 'HBase' and 'Cassandra' for storing the data in the serving layer as most queried data is stored in this layer. These storage solutions offer fault-tolerant and scalable storage and management. The next section provides different case studies of Big Data Architectures found in literature.

2.2.4 Case studies: Big Data Architecture

The following are architectures developed for solving Big Data related problems. Each of these implementations shares a common goal; to analyse large amounts of data as quickly as possible, but making use of different tools depending on the application to achieve this.

- Case study 1:

  The first example is from Marchal et al. (2014), who developed the architecture seen in Figure 2.5 to provide a means of network security monitoring on local enterprise networks.

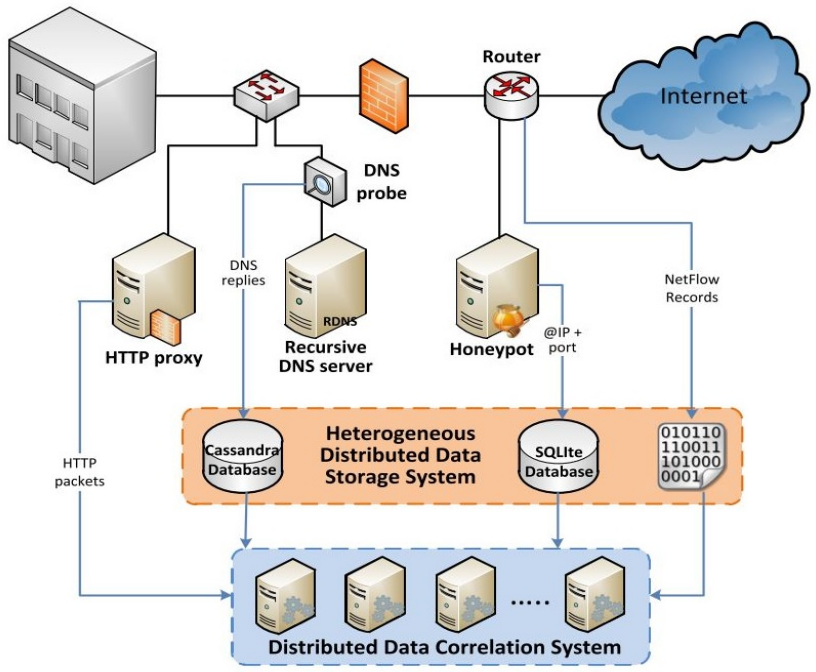


Figure 2.5: The architecture developed by Marchal et al. (2014) in order to provide a large-scale security monitoring system.

Marchal et al. (2014) divided the architecture into two systems; the first provides the data storage and management component of the architecture, while the second system is responsible for data exploitation. The exploitation system collects DNS (Domain Name Servers) data, NetFlow (protocol

which collects IP traffic data) records, HTTP (The Hypertext Transfer Protocol) traffic and IP (Internet Protocol) data, which is the number used to identify an unique network hardware component.

Referring back to Figure 2.5, the architecture collects data from the internet, the DNS probe relays all DNS packets to the 'Recursive DNS server' which extracts the information and stores the data on a Cassandra database. HTTP traffic is monitored using *stream processing*, for any malicious connection attempts by the 'HTTP proxy' and 'Distributed Data Correlation System'. 'NetFlow' monitors the traffic through the enterprise routers and detects intrusions made. Records of communications between routers are stored on the 'Heterogeneous Distributed Storage System' for future analysis to identify any malicious intent. Machines emulating the network are added to attract and detect any malicious user on the network. The user would try to gain access to these 'Honeypot' machines, and the data of 'Honeypots' are stored and analysed to prevent future attacks. All this data is stored on a SQLite database known as 'Dionaea'.

- Case study 2:

  The architecture developed by Xhafa et al. (2015) is considered next. The architecture is developed in order to provide a framework which can be used in the development of Smart City computer applications. The architecture is shown in Figure 2.6 and is based on the *Lambda Architecture*.

  New data is received and sent to the *batch* and *stream* processing layers. In the *batch* layer, the 'preprocessing' is where data is cleaned, features extracted, and filtered to be used in the 'model learning' stage. During the 'model learning' stage, algorithms and statistical analysis are conducted on the data received by devices used in smart city applications.

  In the *serving* layer, the 'feature view' is used to store all pre-processed feature values so they can be quickly accessed. The 'model view' stores all model parameters used in the 'model learning' stages mainly to be used in the *speed* layer. Therefore the applications can store relevant parameters

and feed in the data to quickly analyse and make adjustments as required by the application.

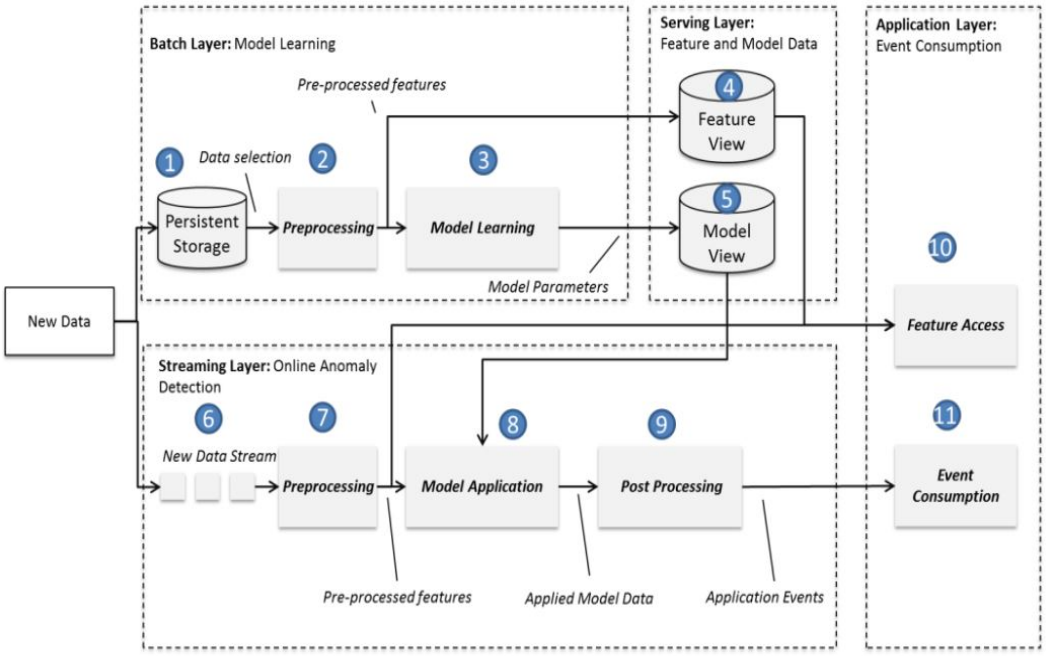


Figure 2.6: The architecture developed by Xhafa et al. (2015) which provides a framework for the development of a Smart City Application.

Similar to the *batch* layer, the *speed* layer takes in the data and conducts 'pre-processing' using equivalent functionality and logic. The 'model application' makes use of known algorithms to conduct anomaly detection in the application or device being used, which is largely the function of the *speed* layer. The 'post-processing' is added to the architecture to detect and remove any false positives (stating an event has occurred when it has not) found in the data, so as to not propagate it further. Finally, the *application* layer serves as means to access real-time views of the data and results found in the *batch* layer. Through dashboards, the 'feature access' can show the anomalies detected in the *speed* layer by the 'event consumption'.

- Case study 3:

The architecture developed by Astakhov and Chayel (2015) is also based around the Lambda Architecture and is a modification thereof for Amazon Web Services (AWS). Amazon provides the infrastructure to companies to conduct cloud-based Big Data Analytics, through this AWS system. The architecture is shown in Figure 2.7.

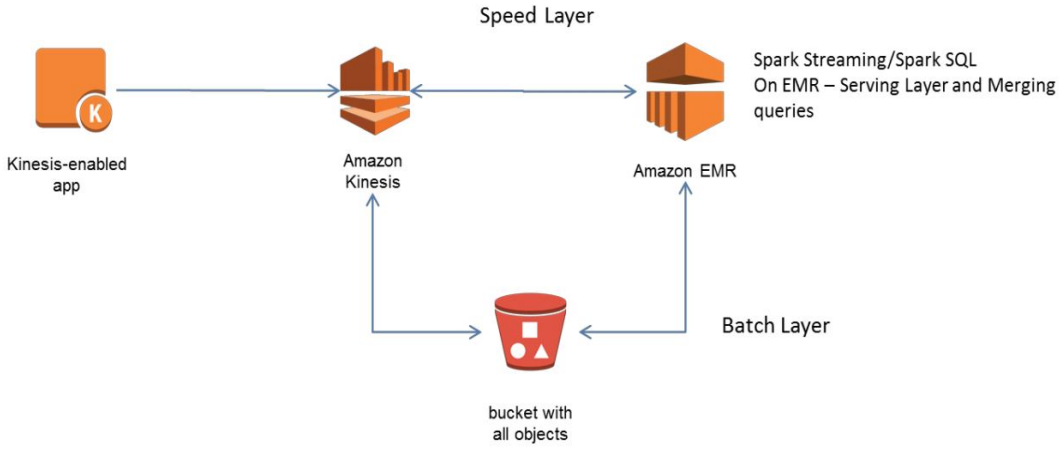


Figure 2.7: The architecture developed by Astakhov and Chayel (2015) according which the Amazon Web Services system is built.

The architecture allows for high-level programming using languages like Java, Python or Scala and is a scalable Hadoop system. The *speed layer* is executed using *Spark Streaming* along with an Amazon EMR cluster (collection of computer nodes) (Astakhov and Chayel, 2015) which provides the analytics and data distribution. The *batch layer* is represented by an Amazon S3 bucket which is the storage solution provided by Amazon to store the company data. The *service layer* is represented by the Kinesis enabled app where the user can view and provide the data to the AWS system from which to run the queries.

- Case study 4:

  The architecture developed by Mayilvaganan and Sabitha (2013) aims to provide a cloud-based Big Data Analytics function for a smart-grid electricity generation system. The architecture shown in Figure 2.8 uses the

Hadoop *Cassandra* project to provide the database, which uses the HDFS format within the framework.

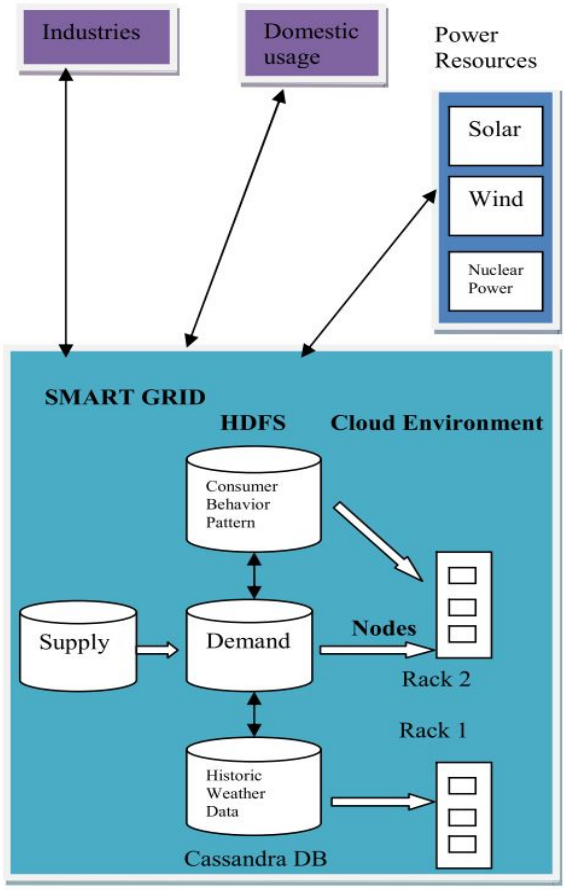


Figure 2.8: The architecture developed by Mayilvaganan and Sabitha (2013) to provide analytics to a smart-grid solution.

The architecture is divided into three high-level entities, the user or query where industry and domestic users are joined to the 'Smart-Grid' entity. This entity conducts the analysis and the data sources or 'Power Resources' which provide the smart-grid with data input. Using information such as demand and supply (current and past) as well as other factors including weather, the data is analysed using the multiple node systems. The data is sent back to the 'industries' and 'domestic' users through a dashboard where queries can be made (Mayilvaganan and Sabitha, 2013). From Mayilva-

ganan and Sabitha (2013), the functionalities include being able to analyse historic weather and predict energy production, analyse user behaviour and predict demand in advance, keeping track of energy production from multiple sources and dynamically switch between sources. Finally, the smart-grid needs to balance demand and supply loads, and efficiently transfer and store the energy.

- Case study 5:

  The final architecture considered is that developed by Li et al. (2017), to analyse the vast amount of data being collected by the civil aircraft industry. The architecture shown in Figure 2.9 is divided into three layers, namely the *IaaS*, *PaaS* and *SaaS* respectively.

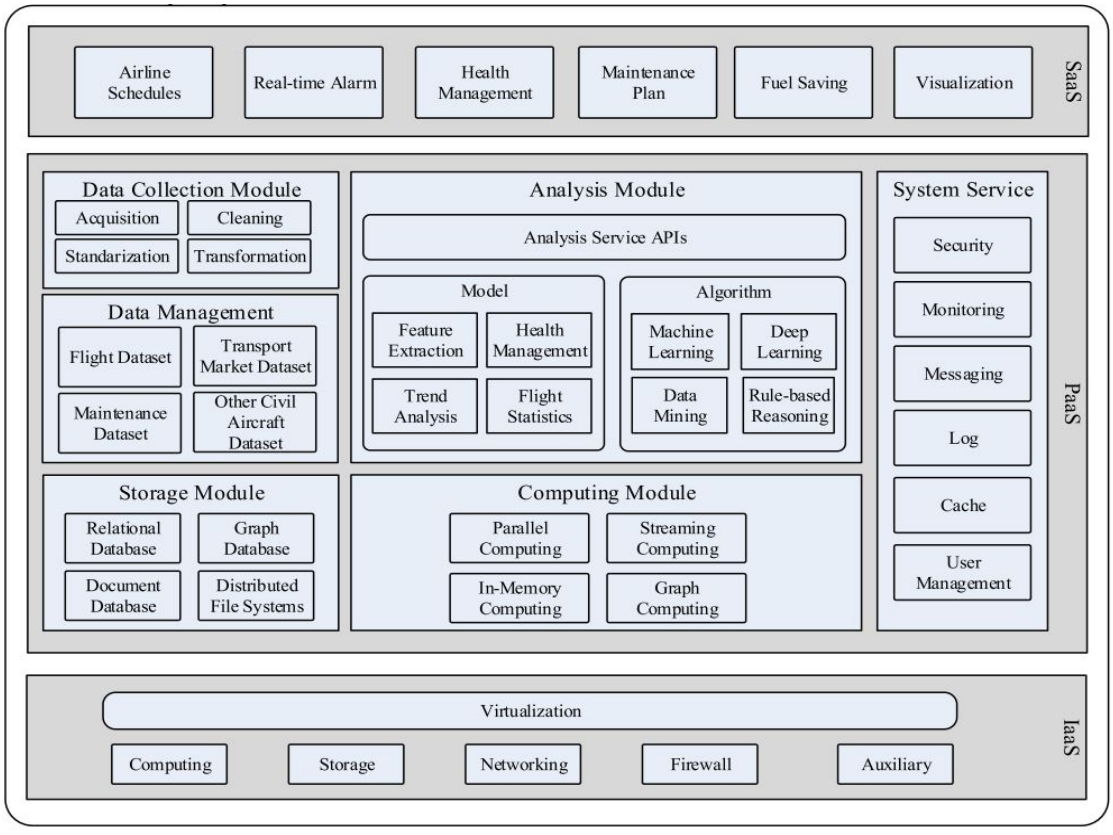


Figure 2.9: The architecture developed by Li et al. (2017) aims to analyse the large volumes of civil aircraft data being collected.

The *IaaS* (Information as a Service) layer according to Li et al. (2017) provides the infrastructure (computer hardware) to develop a distributed virtual computer cluster on which the *PaaS* (Platform as a Service) software is deployed. The *PaaS* as stated in Li et al. (2017) provides the modules which run on the multiple clusters, these include the *Data Collection*, which collects and conducts pre-processing on the incoming data.

Data Management organises the data into different datasets from the different sources, while the *Storage* is concerned with the storage of the data on the HDFS and relational file systems. The *Analysis* makes use of Big Data Analytics discussed further in section 2.3, to provide the desired results. The *System Service* ensures security and efficient running of the system, while the *Computing* modules are tasked with distributing the data across the multiple nodes.

The final entity, *SaaS* (Software as a Service) is similar to the *service layer* of the Lambda Architecture where the user is provided with a visualisation of the data through a 'dashboard' on which to run queries. According to Li et al. (2017), applications for this system include airline schedules that can be analysed or created, while providing a real-time alarm if aircraft thresholds are exceeded. The platform also provides a means to monitor aircraft, create a maintenance plan, and analyse routes for fuel savings.

2.2.5 Summary of Big Data Architecture

This section started by identifying what is understood to be an *architecture*: it is a set of structures that together fulfil an objective to achieve a specific goal (Maier, 2013). A *software architecture* is developed from principal design decisions that form a system for a given application (Taylor et al., 2010). A *reference architecture* is an abstract platform developed to aid in the design process which contains components that were deemed important and require further research and detailed definition. Failing to identify the stakeholders of the architecture and as a whole the project stakeholders, in order to guide the development and perform managerial support, was highlighted as an omission made in industry.

Next, a Big Data reference architecture was researched which would provide a set of requirements that can be applied in the design and development of this project and its architecture (Maier, 2013).

Following on from this, a detailed overview of the different components included in the reference architecture was provided.

A specific reference architecture, the *Lambda Architecture*, was then researched after concluding the research on a Big Data *reference architecture*. The reason for choosing this architecture is because of its current popularity, simplicity and ability to provide results in a timely, accurate manner. The architecture is able to provide these attributes whilst being fault tolerant (Marz and Warren, 2015).

The *Lambda Architecture* as discussed is made up of three high-level components, the *Batch layer*, *Speed layer* and the *Serving layer*. Within each of these the different properties were identified, as well as hardware and software requirements that needed to be included to fulfil the requirements of the architecture.

Finally, different architectures were researched and discussed, some being developed either making use of the *Lambda Architecture* and/or including various components of the Big Data *reference* architecture. The argument made for including these studies is to illustrate the use case and applicability of this architecture to this thesis project.

2.3 Big Data Analytics

An analysis and discussion of current analytics methods is carried out in this section, with the goal of identifying current methods available and then applying the these methods to the Big Data Analytics Demonstrator developed. Russom (2011) describes Big Data Analytics as the application of advanced analytics techniques for business intelligence (BI).

Previously, before the advent of Big Data, small samples of data had to be used to aid decision-making when performing statistical analyses of data. With the advent of Big Data, larger volumes of data could now be analysed, increasing the accuracy of results and predictions. This in turn allows for better decision-making as outlined by Russom (2011).

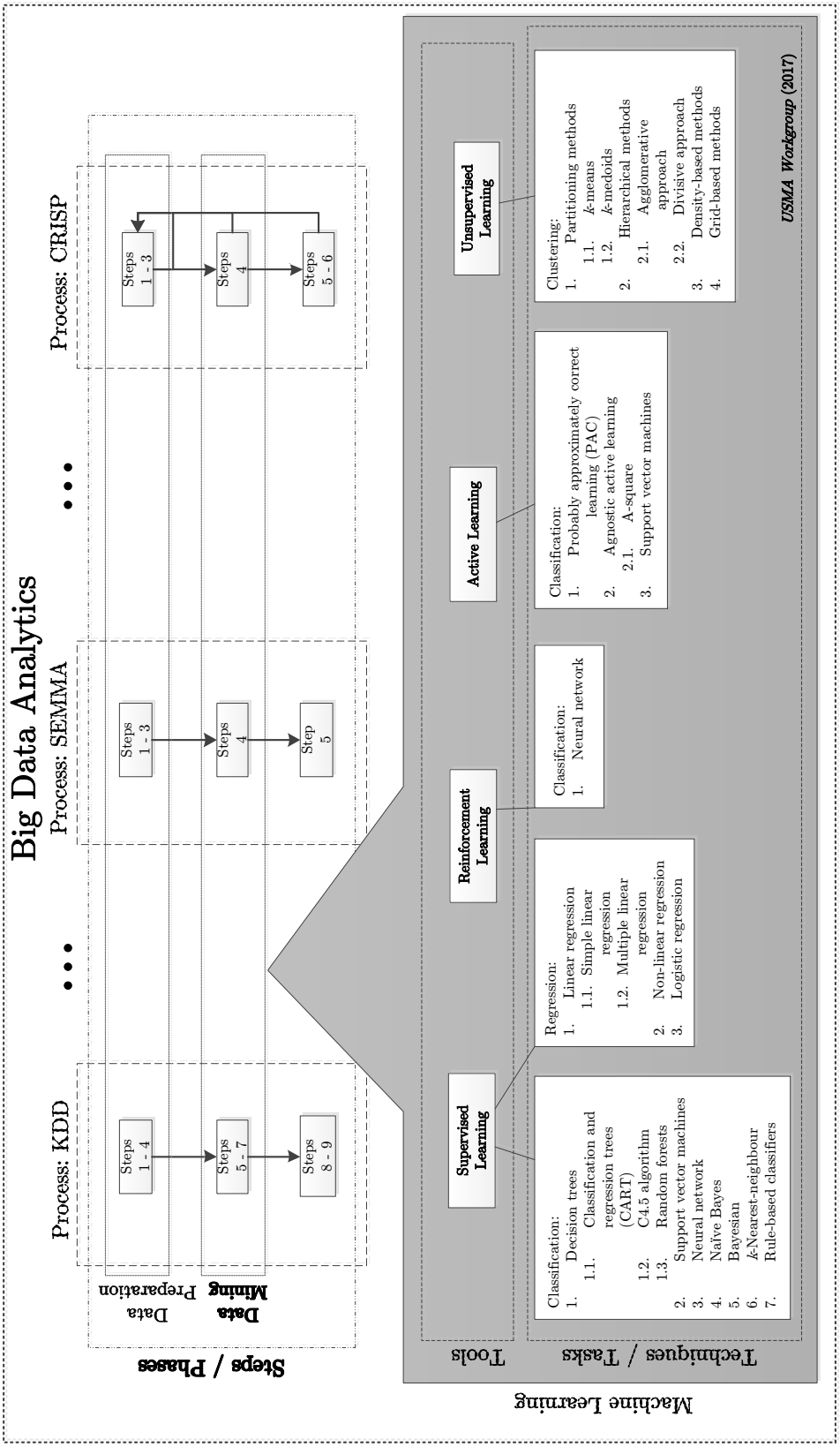


Figure 2.10: The different methods, phases and techniques involved in conducting Big Data Analytics (USMA, 2017).

What follows in this section is a detailed look into what components are defined under 'Big Data Analytics'. Working together in a focus group, members of the USMA research group at Stellenbosch University's Industrial Engineering department developed an understanding of Big Data Analytics, shown in Figure 2.10. The figure shows that Big Data Analytics consists of a process, of which there are three well-documented processes, discussed further. Within each process, there is a data mining step, where machine learning tools and techniques are applied to analyse the data. Following these processes a successful analysis could be conducted. After discussing these processes further, a brief overview follows on *Data Mining*. The tools and techniques known as *machine learning*, shown in Figure 2.10 to analyse Big Data, are then considered. The basic concepts of the most frequent analytic techniques found in literature are discussed, each in a separate sub-section. The techniques pertaining to *active learning* are, however not discussed, but are shown in the figure for completeness.

2.3.1 Analytics process

The three analytics processes discussed in this section are the *Knowledge Discovery in Databases* (KDD), (*Sample, Explore, Modify, Model and Assess*) (SEMMA) and *Cross-Industry Standard Process for Data Mining* (CRISP) process. Using Witten et al. (2016), Jiawei et al. (2012), Azevedo and Santos (2008) and Mariscal et al. (2010), the steps/phases outlined in each process were identified. There are other analytic process methodologies, but to focus the project, the three most widely used and recognised processes are discussed.

- Knowledge Discovery in Databases (KDD): Feyyad (1996) describes the discovery of patterns and trends existing in data using different terms, data mining, data extraction for the acquisition of knowledge and information discovery. There is agreement, however, that data mining is considered to be a phase that is undertaken in the pursuit of this knowledge or information in the analytic processes of SEMMA and CRISP (Feyyad, 1996) and (Jiawei et al., 2012) . This definition and understanding of data mining was made in Butler and Bekker (2017) and is also applied in this project. If the steps outlined in an analytics process are not followed before performing

data mining, "data dredging" can occur as stated in Feyyad (1996), which results in acquiring useless information from the data.

1. Developing and understanding of the application domain, the relevant prior knowledge, and the goals of the user.
2. Creating a target dataset: This involves selection of a dataset, focusing on a subset of variables or data samples to perform data discovery.
3. Data cleaning and pre-processing: During this phase, noise is removed from the data, missing information is added and time sequence information and known changes are accounted for.
4. Data reduction and projection: Finding useful features to represent the data with, depending on the goals. Dimensionality reduction and transformation methods are conducted.
5. During this phase, the matching of KDD process goals to data mining method(s), regression and classification, *etc.* is conducted.
6. Choosing of the data mining algorithm(s): During this phase, the tools shown in Figure 2.10 are chosen, along with the relevant technique(s). This means also matching a technique with the overall criteria of the KDD process.
7. Data mining: The searching of patterns that are of interest using the techniques like classification, regression and clustering.
8. Interpretation of the patterns discovered, whereby if required iteration of the previous phases (1 – 7) can be performed if required. During this phase, visualisation of the results and models is provided.
9. Consolidating discovered knowledge: This phase involves incorporating the knowledge gained into other systems, documenting the results, and reporting. Finally, conducting checks to ensure conflicts between knowledge gained and what was previously known knowledge are resolved.

- Sample, Explore, Modify, Model and Assess (SEMMA): The SEMMA (Sample, Explore, Modify, Model and Assess) methodology was developed and

integrated into by the SAS institute into its tools, therefore it is typically applied to enterprise miners (a tool used to conduct data mining) (Mariscal et al., 2010). The methodology provides a means to understand, organise, develop and maintain a data mining project. The KDD process is however applicable to a wider array of applications and environments. The SEMMA process consists of five phases, as follows:

1. Sample: Data samples are collected from large datasets. These are used to draw significant insight and results from, but are of a manageable size to allow for quick and easy manipulation.
2. Explore: During this phase, data exploration includes searching for anomalies and trends to better understand the data and allow for idea generation later on during the modelling phase.
3. Modify: The data is created, selected and transformed to allow for a more focused modelling selection process. This includes identifying outliers and conducting variable reduction (Shafique and Qaiser, 2014).
4. Model: The sampled and modified data are now used during the modelling phase, where the software model automatically uses the tools and techniques shown in Figure 2.10 to search for combinations of data to be used in predictions (Azevedo and Santos, 2008).
5. Assess: The results and findings of the analysis are evaluated for usefulness and reliability from the data mining process, thereby establishing how well the model has performed.

- Cross-Industry Standard Process for Data Mining (CRISP): This process methodology was developed by a consortium from Daimler, Chrysler, SPSS and NCR (Azevedo and Santos, 2008), providing an uniform framework along with guidelines for performing a data mining activity as stated by Shafique and Qaiser (2014). The process consists of the following six phases:

  1. Business understanding: During this phase, relevant factors such as success criteria, business and data mining objectives are established. Terminologies and technical terms are also defined.

2. Data understanding: Data collection is conducted whereby data quality is checked. Thereafter data exploration is conducted to gain insight which will assist in the formation of a hypothesis which will guide the analytics process to uncover hidden results or insights.
3. Data preparation: The final dataset is selected and prepared for the modelling phase. This includes table and attribute selection, task records and cleaning and transforming the data.
4. Modelling: The relevant application and tools and techniques as seen in Figure 2.10, are selected to best test the hypothesis and fulfil business and data mining objectives. Parameters are set, and various models are built for the same problem, which is used as an evaluation method.
5. Evaluation: Each of the models built is evaluated and the results of the models are interpreted to determine if the desired objectives were met.
6. Deployment: In this final phase the decisions are made based on the knowledge and results obtained from the models. The results and knowledge are then organised, reported and presented as required.

The SEMMA process was not considered for the analytics component of this project as it has limited applications and is linked to the SAS enterprise miner software. The CRISP methodology Azevedo and Santos (2008) was concluded to be an implementation of the KDD process. Following from this, the next section briefly discusses the various machine learning tools and techniques that are used in all three of the processes shown in Figure 2.10.

2.3.2 Data Mining

Cabena et al. (1998) define Data mining to be "The process of extracting previously unknown, valid, and actionable information from large-scale databases and then using the information to make crucial business decisions." Another definition of data mining given by Jiawei et al. (2012) of data mining is "A process of discovering interesting patterns and knowledge from large amounts of data."

Cabena et al. (1998) state that the need for data mining and the means by which to implement it have driven research and development in searching for valuable information in order to make informative and data driven-decisions. Drivers of data exploration are:

- Focus on the customer: This driver's purpose is to generate customer intimacy, collaboration and having a one-to-one partnership.
- Focus on the competition: Identifying what competitors are doing, and develop and change to ensure a competitive advantage.
- Focus on the data asset: The potential value in the ever-growing amount of data being stored by organisations.

The following is a list of enablers identified by Cabena et al. (1998) that together with the drivers, allow for data mining and data-driven decision-making:

- Data flood: Lower data storage costs and higher storage capacity have allowed organisations to store an ever-growing amount of data to be mined for insights.
- Growth of data warehousing: Cleaned and documented datasets kept by organisations with the computing power available have allowed for data mining.
- New information technology (IT) solutions: Lower cost IT solutions and processing capability allow for large-scale data mining. An example of a fast parallel method of processing large datasets is Hadoop, which is an open-source platform that takes in large datasets and transforms them into a Hadoop Distributed File System (HDFS) format. The data is stored and processed across multiple computers (each computer is called a node), each having a storage and processing component.

In order to successfully conduct a data mining exercise, Cabena et al. (1998) divide the data mining process into various steps. The steps outlined in Cabena et al. (1998) follow a very similar pattern to that discussed in section 2.3.1 where

there is a *Data Mining* stage that is conducted. Along with this, as discussed in section 2.3.1, data mining is a step in the anlytics process and makes use of *machine learning* tools and techniques in order to mine the data for results (Witten et al., 2016), discussed in further detail in section 2.3.3.

As stated in Cabena et al. (1998), this step "amounts to running the algorithms" on the gathered data that is typically pre-processed in previous phases of the analytics process, previously discussed in section 2.3.1. Which *machine learning* tools and techniques are used and what is required from the data mining varies and is dependent on the application or intent of the analyst. To store the data that is collected and pre-processed, *data warehouses* can be used by an analyst, company or organisation for future use (if desired). A data warehouse allows for the data that has also been through the data mining step to be stored for historical purposes. Jiawei et al. (2012) define a data warehouse to be a "repository of information collected from multiple sources, stored under a unified schema" and according to Cabena et al. (1998), assists in supporting management decisions.

Data warehousing, although an enabler to data mining ventures, is not a prerequisite to a data mining solution. Data warehousing allows for more effective data-driven decision-making. A data warehouse is seen as a neutral holding area for quality data used for decision making (Cabena et al., 1998). Mining the data stored in a data warehouse involves different analytical techniques and is done so as to improve an organisation's decision-making. All the processes, techniques and tools involved in decision-making are defined as Business Intelligence (BI).

The research done by Jiawei et al. (2012), found that data warehouses can be modelled through a multidimensional data structure known as *data cubes*, with each dimension representing a(n) attribute(s) and each cell storing a value. Given in Figure 2.11 is a three-dimensional representation of a data cube, but the data structures can contain more than three dimensions (which cannot be visually represented).

Provided in Table 2.3 are the tools used in data mining, with each tool discussed in a separate section of Gera and Goel (2015). The frequently implemented tools and techniques are thereafter discussed further in section 2.3.3 of this thesis.

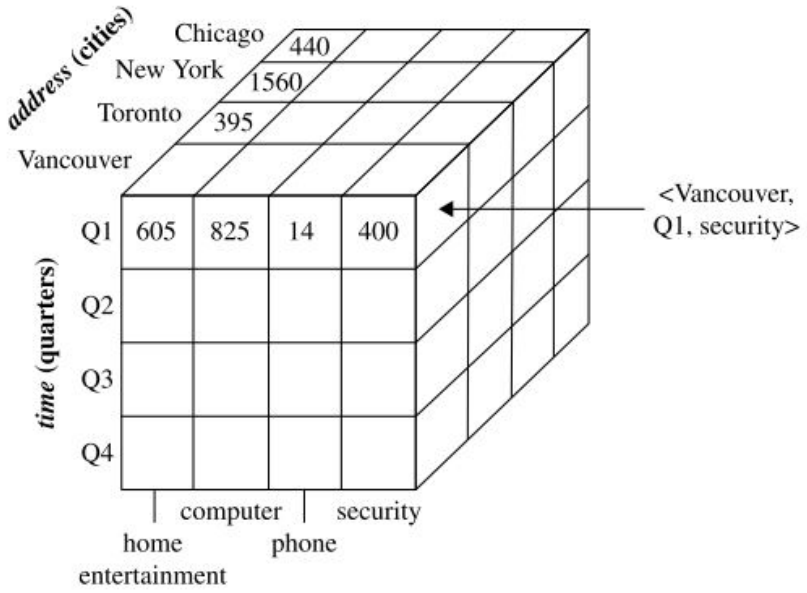


Figure 2.11: A visual representation of a data cube, with multiple dimensions and a value within each cell (Jiawei et al., 2012).

Ethical issues that arise from conducting data mining according to Witten et al. (2016) also need to be considered. Dependent on the application, certain data needs to be removed as to prevent companies, analysts or organisations from using the data to discriminate. Examples given are whether a person receives a loan or not dependent on irrelevant factors such as race or gender, often included in the data gathered for data mining activities. Such data can however be relevant, as stated in the example given in Witten et al. (2016), but supervision is required and anonymisation where necessary needs to be implemented before an analyst, organisation or company receives the data.

Next, the frequently used and most relevant tools and techniques that are available for data mining shown in Figure 2.10 are discussed further.

2.3.3 Machine learning

As stated in Butler and Bekker (2017), various techniques from multivariate statistics, data mining, and advanced predictive analytics, are performed to generate predictions or identify patterns in data. These techniques are widely known

Table 2.3: Collection of software tools available for conducting data mining, each with a particular objective dependent on the requirements of the data mining activity (Gera and Goel, 2015).

Tool	**Objective**
Orange	Visual data analysis
WEKA	General machine learning package
Kernlab	Kernel based classification and dimensionality reduction
Dlib	Portability
Nieme	Linear regression, classification
Java-ML (Machine Learning)	Feature selection
pyML	Kernel methods
Shogun	General purpose ML package, focus on large scale learning
Mlpy	Basic algorithms
Torch7	Neural networks
Pybrain	Reinforcement learning

as machine learning (ML). Machine learning is considered to be the application of algorithms to allow for autonomous learning of a system, or as stated, the ability to perform an analysis without the need to be explicitly programmed to fulfil the desired task, by Talegaon (2014). Applications of machine learning range from identifying user preferences, providing recommendations, identifying spam and non-spam emails to the probability of winning a lottery.

Using Figure 2.10 as a guide, the most frequently used and relevant techniques are discussed further under each tool other than *active learning*. These tools and techniques as previously mentioned assist in the analytics process when conducting data mining.

Machine learning is divided into three methods (Ayodele, 2010), namely:

- Supervised machine learning
- Unsupervised machine learning

- Reinforcement learning.

Supervised machine learning involves algorithms which learn to best fit what is known regarding the input data with the known output data. Predictions are then made for future data outputs given the input training data as described by Ayodele (2010). In supervised ML, *regression* and *classification* techniques are used.

Provided with a dataset that contains data which is numerical and continuous, regression techniques are an applicable method to make predictions. A prediction is made when the relationship or factor between the dependent variable and independent variables is established which reduces the error in those predictions.

The three most common regression techniques currently available are Linear, Non-Linear and Logistic Regression. To improve how these regression model fit the input data, Berson et al. (2005) provides four additions that can be made. These additions or modifications are: a) Adding more predictors, b) Applying transformations to predictors, c) Multiplying predictors together or d) Modifying the predictors to accommodate binary (yes/no, 1 or 0) outcome values, otherwise known as Logistic Regression.

When working with categorical data, common groupings have to be formed in order to make predictions. Classifications, which groups data of different types into a given class, allow for this.Kotsiantis (2007) outlines the two groups of classification algorithms. The first is *Artificial Intelligence* which uses perception-based techniques and the second is *Statistics*, where Bayesian Networks and instance-based techniques are used. Examples of logic-based techniques are Decision Trees (DT), and perception-based techniques are Artificial Neural Networks (ANN). Instance-based techniques are for example, *Lazy Learners* or *Nearest Neighbour*.

When working with data from which no knowledge of the output or the effect on the output is known, the data is considered unlabelled (Ayodele, 2010). If the data is unlabelled, unsupervised machine learning techniques are applicable. These techniques group observations or data points together in clusters, identifying possible patterns or trends through this. These clusters are then evaluated, using inference to determine if the clusters best describe the input data. As stated in Jain et al. (1999), clustering classifies patterns (observations) into

groups. Properties typically of concern in clustering are 1) whether or not the algorithm can handle a type(s) of attributes, 2) scalability to larger datasets, 3) the handling of outliers, 4) finding irregularly shaped clusters, 5) data order dependency (Kogan et al., 2006).

Reinforcement learning is a sequence of correct actions that are assessed for goodness by learning from a series of 'bad actions' taken, in order to reach a goal. The algorithm thus learns from incorrect actions and generates the correct actions that need to be taken in future Alpaydin (2010). Barto and Mahadevan (2003) define active learning to be a method by which an agent learns to approximate the optimal behavioural strategy whilst interacting with the environment the agent is in. Reinforcement learning is therefore considered to be conditioning or condition learning, and according to Maia (2009) can be divided into two categories namely, *classical* and *instrumental*. The difference is that in *classical*, the outcome does not depend on the actions of the agents. In *instrumental*, where an agent, given a positive response (R) from a stimuli (S), the connections are strengthened and the system is rewarded. The opposite holds true given a negative response. The goal as given by Maia (2009) is to learn which actions to select to maximise long term reinforcements. The elements fundamental to a reinforcement learning algorithm are the states, actions and reinforcements the system is in or rewarded.

The goal of understanding the different machine learning techniques is to then use these various techniques during the model development and implementation stage of the BDAD. The techniques will be implemented so as to provide automated analysis of the large datasets, to extract meaningful insights and ultimately provide BI as set out in the project proposition.

2.3.3.1 Supervised learning: Regression

From Alpaydin (2010) a basic representation of a model/function to represent supervised ML is

$$y = g(x|\theta), \tag{2.1}$$

where $g()$ is the function which describes the input data to the output data, and θ is the parameters of the data points. The y is the desired output for a regression or class code (*e.g.* in a binary classification either 0 or 1) (Alpaydin,

2010). The goal of the ML algorithms is to optimise θ such as to reduce the 'approximation error' between the function and input data, the net result being the algorithm would then best match the training data and provide a continually improving result. A short summary of the different *regression* techniques, their applications, and sources where each is discussed is given in Table 2.4, as taken from the focus group held with other members of the USMA research group. Some of the sources listed are used in the discussions that follow, while others are added for completeness. Next, the different techniques used in regression are discussed.

Table 2.4: Summary of regression techniques.

	Regression Technique:	*Source:*	*Known for/Application:*
1	**Linear Regression**	Salkind (2007) Gera and Goel (2015) Paliwal and Kumar (2009) Salkind (2007)Yang et al. (2017) Seber and Lee (2003)	A model that can show relationships between two variables and how the independent variable impacts the dependent variable.
1.1	Simple Linear Regression	Bishop (2013) Paliwal and Kumar (2009) Salkind (2007)	Evaluate trends Forecasting Analyse marketing effectiveness Assess finance/insurance risks
1.2	Multiple Linear Regression	Bishop (2013) Paliwal and Kumar (2009) Salkind (2007)	Same as Simple Linear Regression, more variables for example in management – Is customer loyalty influenced by customer satisfaction, brand perception and price perception?

Continued on next page

	Regression Technique:	*Source:*	*Known for/Application:*
2	**Non-Linear Regression**	Bates and Watts (1988) Chatterjee and Hadi (2006) Gallant (1975) Gera and Goel (2015) Riffenburgh (2011) Ruckstuhl (2010) Tellis (2006) Tellis and Ambler (2007)	Assesses the effects of wear-in and wear-out on campaigns. This can be done by determining the effectiveness of advertisements on different age groups.
3	**Logistic Regression**	Chatterjee and Hadi (2006) Hosmer and Lemeshow (2013) Karp (1998) Montgomery et al. (2012) Riffenburgh (2011) Salkind (2007)	Assesses likelihood of remaining a customer Sales – purchase/re-purchase vs. No purchase Marketing – Respond vs. Not respond Fraud identification Health care – Cure *vs.* No cure

- Linear and non-Linear Regression:

  The simplest regression model is a linear regression model with two variables, where the desired relationship is derived from testing one dependent variable (y_i) against independent variables (x_i) if there are n pairs of (x_i, y_i) ($i = 0, 1, 2...n$) variables and a straight line is fitted to the model (Seber and Lee, 2003). A user can then use this linear relationship to make predictions based on the linear function (noting that the linear function is an approximation and contains a certain degree of error, and is not an exact relationship between the dependent and independent variables.) Some relationships are however non-linear. After discussing linear regression and parameter estimation methods within linear regression, non-linear regression is discussed. To focus the literature around regression, for linear and non-linear, only the model and methods around parameter estimation will be discussed, as after this the model/function is complete. Firstly, what follows is a mathematical model of linear regression:

  A dependent variable is given by Y and independent variables are given by $X_1, X_2, ..., X_K$, the model is then given by

  $$E[Y|X_1 = x_1, X_2 = x_2, ..., X_K = x_K] = \phi(x_1, x_2, ..., x_K), \tag{2.2}$$

  from this, the linear regression equation is given with various parameters (β) which are estimated in order to fit the linear function to the variables with the minimum error between the variables and the function,

  $$\phi(x_1, x_2, ..., x_K) = \beta_0 + \beta_1 x_1 + ... + \beta_K x_K. \tag{2.3}$$

  By varying the parameters β, the model can be adapted to best fit the test data and provide the best prediction possible. In Figure 2.12, a linear function is fitted to the data resulting in the equation shown. For this example, the dependent variable (Y) represents the cost ($) of a domestic airline ticket in the United States and the independent variable (X) is the distance (miles). The parameters (β) are $\beta_0 = 109.95$ and $\beta_1 = 0.0505$

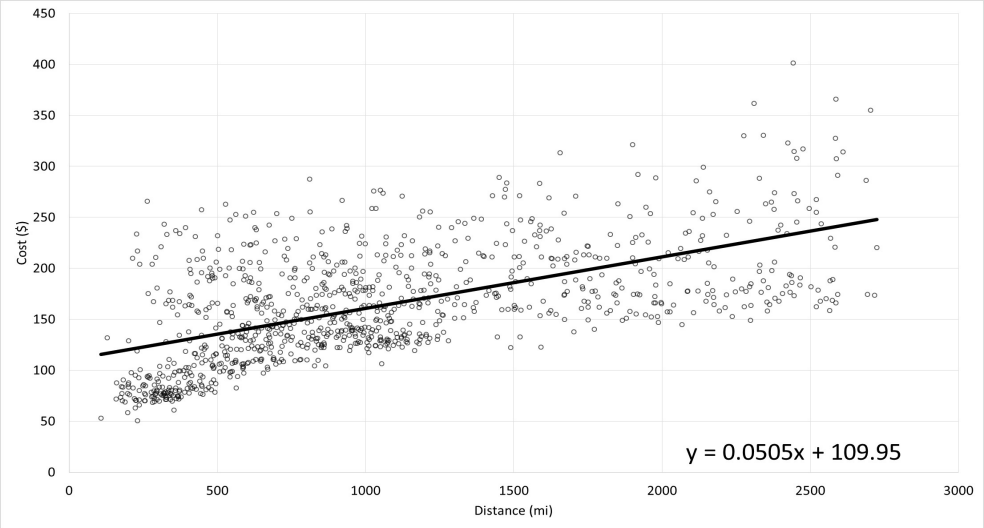


Figure 2.12: An example of a linear regression fitted to a dataset, showing the relationship between the cost of an airline ticket and the distance travelled. The dataset was obtained from Association (2016a).

which can be estimated, given that the parameter values are not known. The linear model using these parameter values is,

$$\phi(x) = \beta_0 + \beta_1 x = 109.95 + 0.0505x, \tag{2.4}$$

given that $K = 1$, as there is only one independent variable.

To estimate the parameter values according to Seber and Lee (2003), the two methods commonly used are the *Maximum Likelihood method* and the *Least Squares method*. Similarly, for non-linear and logistic regression, the *Maximum Likelihood method* and the *Least Squares method* are used to estimate the parameter, the derivations of those functions can be found in Harrell (2015), Seber and Lee (2003), Montgomery et al. (2012), Ruckstuhl (2010), Wild and Seber (2005), Hosmer and Lemeshow (2013) and Menard (2002). Only the final function of each estimation method is given for illustrative purposes as the focus is not on the function's derivations, but rather the output the functions provide.

Maximum likelihood:

Harrell (2015) denotes a sample size of n with each sample having an observed response (dependent variable) of Y_1, Y_2,...,Y_n and independent variable x_i $(i = 0, 1, 2...n)$. From Seber and Lee (2003) in order to determine the parameter values, let

$$\begin{aligned} Y &= \theta_i + \epsilon_i \\ &= \beta_0 + \beta_1 x_i + \epsilon_i \end{aligned} \tag{2.5}$$

if the following assumptions from Seber and Lee (2003) of the elements of ϵ_i are made:

- it is unbiased,
- has a constant variance,
- is uncorrelated and,
- is normally distributed

where the distribution error ϵ_i is independently distributed along a known normal distribution $N(0, \sigma^2)$. As stated by Harrell (2015), with significantly large sample sizes the variance σ^2 can be approximated by a normal distribution. For this reason when calculating the variance in the *maximum likelihood* method the variance will be assumed to be normally distributed. The joint probability or likelihood function (function that describes the probability of the response occurring) of the observed data would then be given by the following (Montgomery et al., 2012):

$$\begin{aligned} L &= \prod_{i=1}^{n} (2\pi\sigma^2)^{-1/2} \exp[-\frac{1}{2\sigma^2}(y_i - \beta_0 - \beta_1 x_i)^2] \\ &= (2\pi\sigma^2)^{-1/2} \exp[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2] \end{aligned} \tag{2.6}$$

The next step involves taking the log of the function in (2.6) as it would allow for parameter estimations where the derivative of the log function is equal to zero. From Montgomery et al. (2012) and using the log of the

likelihood function, the parameters $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma^2}$ (biased estimator of σ^2) are estimated as to maximise L. Montgomery et al. (2012) then provide the solution to the parameters to be the following for β_0, β_1 and σ^2:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2.7}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} y_i(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.8}$$

$$\hat{\sigma^2} = \frac{\sum_{i=1}^{n}(y_i\hat{\beta}_0 - \hat{\beta}_1 x_i)^2}{n} \tag{2.9}$$

Using these parameter values, the best linear fit to the test data can be achieved. This linear function could then be used to make predictions with a certain error. The example function from Figure 2.12 had parameters estimated which provided the following regression model (function),

$$\phi(x) = 0.0505x + 109.95 \tag{2.10}$$

which can be used to make future predictions of the cost of a flight depending on the distance required to travel, with a certain error ϵ_i.

Following from the *maximum likelihood method*, the *least squares method* is another method considered in linear regression as it allows for parameter estimation regardless of the distribution of errors ϵ_i, Seber and Lee (2003).

Least squares:

Similar to what was used in the *maximum likelihood* method, suppose there are n pairs of test data $(y_1, x_1), (y_2, x_2), ..., (y_n, x_n)$. Using this data, parameters β_0 and β_1 are to be estimated. The goal of the *least squares method* is to estimate these parameters in order to minimise the difference between the distance of the straight line regression and the data points. The model

that uses these parameters therefore represents a 'near optimal' (with a certain error) solution to the test data. The model could be used in predictions as it represents the trend followed in the test data. The method of least squares is described using Seber and Lee (2003) and Abdi (2007) as both have similar steps. The notation of Seber and Lee (2003) will however be used as it is similar to that used in the description of the maximum likelihood method. Using (2.5), the least squares equation is,

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \tag{2.11}$$

similar to the *maximum likelihood method* the derivative is solved by setting it equal to zero for each parameter β_0 and β_1, known as the normal equations. Solving the normal equations yields

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2.12}$$

and for β_1,

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i\right)}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}} \tag{2.13}$$

where

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{2.14}$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{2.15}$$

The error or residual as stated by Seber and Lee (2003) would then be

$$\epsilon_i = y_i - \hat{y}_i = y_i - \left(\hat{\beta}_0 + \hat{\beta}_1 x_i\right). \tag{2.16}$$

Non-linear regression is considered next, because few relationships between dependent and independent variables exhibit exact linear relationships. Firstly, the non-linear model description is given, then similar to linear regression, the methods to estimate the model parameters will be discussed. The notation used in the linear regression model discussion will be used here in the non-linear discussion, using Wild and Seber (2005) and Ruckstuhl (2010) as both have similar generic descriptions of a non-linear model and methods for parameter estimation.

The non-linear model form is given in Wild and Seber (2005) by

$$y_i = f(\boldsymbol{x_i}, \boldsymbol{\beta_i^*}) + \epsilon_i \qquad (i = 1, 2, ..., n), \tag{2.17}$$

where $\boldsymbol{x_i}$ is a $k \times 1$ vector of independent variables, $\boldsymbol{\beta_i^*}$ is a $p \times 1$ vector of parameters. To illustrate a non-linear relationship, the following example of the speed and fuel consumed during shipping operations between 2003 and 2006 through a Taiwanese port will be used.

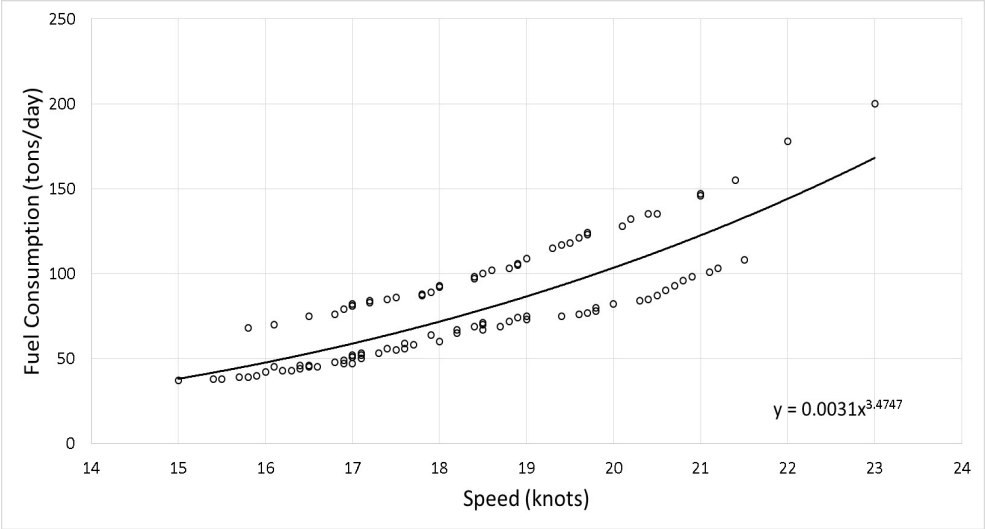


Figure 2.13: An example of a non-linear regression fitted to a dataset, showing the relationship between the speed of ships and fuel consumed around a Taiwanese port. The dataset was obtained from Association (2016b).

From the regression function in the example, it can be noted that the non-linear model of the test data is given by,

$$\phi(x_1) = \beta_0 x_1^{\beta_1} \tag{2.18}$$

with $\beta_0 = 0.0031$ and $\beta_1 = 3.4747$. To determine the parameter values, both *maximum likelihood* and *least squares* method are considered and used in Wild and Seber (2005) and Ruckstuhl (2010) for non-linear models.

Maximum likelihood:

$$p(y|\boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} \left[\sigma^{-1} g\left(\frac{y_i - f(\boldsymbol{x_i}, \boldsymbol{\beta})}{\sigma} \right) \right] \tag{2.19}$$

If ϵ_i is $N(0, \sigma^2)$ then,

$$p(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2} \sum_{i=1}^{n} \frac{[y_i - f(\boldsymbol{x_i}; \boldsymbol{\beta})]^2}{\sigma^2} \right), \tag{2.20}$$

with the log of the likelihood function taken to maximise (2.20) and as before, taking the derivatives at zero for estimating the parameters β_0 and β_1.

Least squares:

The least squares formula for non-linear regression is similar to that used in (2.11) but takes into account the function f in (2.17) to give

$$S(\beta) = \sum_{i=1}^{n} \left[y_i - f(\boldsymbol{x_i}, \boldsymbol{\beta}) \right]^2, \tag{2.21}$$

with ϵ_i independently and identically distributed (N(0,σ^2)) with variance σ^2. After following derivations given in Wild and Seber (2005) and Ruckstuhl (2010), the least squares equation for all $\boldsymbol{\beta}$ can be expressed as

$$S(\boldsymbol{\beta}) = ||y - f(\boldsymbol{\beta})||^2. \tag{2.22}$$

Finally, logistic regression is considered, where similar to the discussion of linear and non-linear regression, first the model is provided and thereafter how to solve for the model parameters.

- Logistic Regression:

  When data is of a binary nature (yes or no, 0 or 1), fitting a model to the test data would not provide meaningful insights. This is where logistic regression can be used. Using Harrell (2015), Hosmer and Lemeshow (2013) and Menard (2002), the logistic regression model, and methods used for parameter estimation will be discussed. In Harrell (2015), the dependent (response) variables are given by Y, where $Y = 1$ indicates that an occurrence has occurred, and $Y = 0$ denotes the inverse thereof. The independent (predictor) variables are given by X, where X is a vector $X = \{X_1, X_2, ..., X_k\}$ of different predictors. Because binary logistic regression has only two outcomes, conditional probabilities are required in order to determine for each response what was the probability of occurring, given a predictor. Therefore, using Hosmer and Lemeshow (2013) for the multiple logistic regression model, each response can be given by

$$P(Y = 1|\boldsymbol{x}) = \pi(\boldsymbol{x}), \tag{2.23}$$

  where $\pi(\boldsymbol{x})$ is a simplified notation of the probability of the outcome given $\boldsymbol{x}$ (predictor). The logistic function given this notation is

$$g(\boldsymbol{x}) = \ln\left(\frac{\pi(\boldsymbol{x})}{1 - \pi(\boldsymbol{x})}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k, \tag{2.24}$$

  and for each $\pi(\boldsymbol{x})$,

$$\pi(\boldsymbol{x}) = \frac{e^{g(\boldsymbol{x})}}{1 + e^{g(\boldsymbol{x})}}. \tag{2.25}$$

After establishing the logistic regression model, the parameters of the vector $\boldsymbol{\beta}$ need to be estimated to complete the model. Similar to linear regression where the *maximum likelihood* method was used by Seber and Lee (2003), Montgomery et al. (2012) and Harrell (2015), the method will be described using Hosmer and Lemeshow (2013), as there are commonalities among all and the notations used are consistent with the model already discussed.

Maximum likelihood:

First, assume there are n observations $(\boldsymbol{x_i}, y_i)$ with $i = (1, 2, ..., n)$. From the logistic regression model there is a vector of parameters $\boldsymbol{\beta'} = (\beta_0, \beta_1, ..., \beta_p)$ with p likelihood equations, so the likelihood equations are

$$\sum_{i=1}^{n}[y_i - \pi(x_i)] = 0 \tag{2.26}$$

and

$$\sum_{i=1}^{n} x_{ij}[y_i - \pi(x_i)] = 0 \tag{2.27}$$

for $j = 1, 2, ..., p$. The solution to these equations is given by $\hat{\boldsymbol{\beta}}$ with values $\hat{\pi}(\boldsymbol{x_i})$. The estimators from Rao (1966), can be obtained using the matrix of the second partial derivatives of the log likelihood function found in Hosmer and Lemeshow (2013).

To illustrate logistic regression, a small dataset containing various factors affecting human fertility was used from Gil and Girela (2013). Table 2.5 shows a subset of the 100 data points in the dataset. To note, the analysis conducted and dataset are for example purposes.

Table 2.5: A subset of data on the factors affecting the fertility of humans (Gil and Girela, 2013).

Season	Age	Childhood disease	Accident	Surgical intervention	High fever	Alcohol consumption	Smoking	Hours sitting	Diagnosis
-0.33	0.69	0	1	1	0	0.8	0	0.88	N
-0.33	0.94	1	0	1	0	0.8	1	0.31	O
-0.33	0.5	1	0	0	0	1	-1	0.5	N
-0.33	0.75	0	1	1	0	1	-1	0.38	N
-0.33	0.67	1	1	0	0	0.8	-1	0.5	O
-0.33	0.67	1	0	1	0	0.8	0	0.5	N
-0.33	0.67	0	0	0	-1	0.8	-1	0.44	N
-0.33	1	1	1	1	0	0.6	-1	0.38	N
1	0.64	0	0	1	0	0.8	-1	0.25	N
1	0.61	1	0	0	0	1	-1	0.25	N

A brief overview of the factors are, during which season the sample was taken, the age of the volunteer, if the person had suffered from a childhood disease, if the person had been in an accident, needed surgical intervention, had a high fever, how much alcohol they consumed, if the person smoked and the number of hours the volunteer spent sitting down. The various factors are scaled as follows:

- Season: Winter = -1, Spring = -0.33, Summer = 0.33, Autumn = 1.
- Age (18-36): Between 0 and 1.
- Childhood disease: Yes = 0, No = 1.
- Accident: Yes = 0, No = 1.
- Surgical intervention: Yes = 0, No = 1
- High fever: Less than 3 months ago = -1, more than 3 months ago = 0, no = 1.
- Alcohol consumption: Between 0 and 1.
    1. Several times a day.
    2. Every day.
    3. Several times a week.
    4. Once a week.
    5. Hardly or never.
- Smoking: Never = -1, occasionally = 0, daily = 1.
- Hours sitting: Between 0 and 1.
- Diagnosis: Normal = N and altered (infertile) = O.

Of the 100 volunteers, 88 were labelled as *normal*, while the other 12 were labelled as *altered*. To simplify the analysis of the example, a logistic regression was conducted using factors *childhood disease*, *Accident*, *Surgical intervention* and *High fever*. Using these factors, the analysis needed to determine whether a person was classified as *normal* or *altered*.

From the analysis, 30 percent of the data was used for training purposes, while the 70 percent was used as the test data. From the test data, 28 were predicted to be *normal*, while 0 were predicted to be *altered*. A variance of 0.88 was achieved, which shows a high approximation by the regression function, however the small dataset size must be noted. A F1-score of 0.9 (from precision value of

0.87 and recall value of 0.93) was achieved, indicating high precision and recall in the prediction. Therefore, the logistic regressions could fairly accurately predict a person to be *normal* or *altered*. The β values from the analysis are as follows, $\beta_0 = -0.25198$, $\beta_1 = -1.0273$, $\beta_2 = -0.2102$, $\beta_3 = -0.8279$.

Conclusion

The discussion of regression has focused on the different types of regression used widely, explained through examples and how the parameters of a regression function can be estimated. Harrell (2015) provides further reading on using the *least squares* and *maximum likelihood* method in linear, non-linear and logistic regression applications. Hosmer and Lemeshow (2013) discuss logistic regression further by looking at applying logistic regression while using different sampling methods, for multinomial or ordinal outcomes and other methods of assessing the fit of the model. The work of Seber and Lee (2003) offers further reading on hypothesis testing in linear regression, interval estimation, analysis of variance, and if assumptions of the models no longer hold, what changes can be made to the model. Next, classification, which is also a form of supervised learning, is considered, along with some of the techniques available.

2.3.3.2 Supervised learning: Classification

Classification exists because there is a desire to group common instances together under a certain class to create logical order. Jiawei et al. (2012) make use of a bank loan officer example to explain the use of classification. In the discussion, the bank loan officer must decide which bank loans are 'safe' and which are 'risky', according to different attributes. The bank can use a classification model (classifier) to determine the likelihood of the loan being classified as either 'safe' or 'risky'. These are categorical labels that can also be represented by discrete values.

Using the bank loan example, a general approach to classification would then be to have a training sample $\boldsymbol{X} = (x_1, x_2, ..., x_n)$ an *attribute vector* with n number of sample points, and attributes $\{A_1, A_2, ..., A_n\}$ for the database. Other attributes can be added such as a persons age, income, *etc.* which would then

be used to again, to classify the different loans according to these attributes. Mathematically, the first part of the classification process can be mapped by the function

$$y = f(\boldsymbol{X}) \tag{2.28}$$

which states that a given class label y can be predicted using sample data $\boldsymbol{X}$. The function (2.28) then separates the data into different classes. The accuracy of the classifier (function) is determined by the number of correctly classified test samples. The classifier can then be used to automatically classify different loans, without the need of the loan officer's intervention.

Following on from the introductory discussion on the different groups of classification in section 2.3.3, a brief overview of logic-based, perception-based, Bayesian networks and instance-based techniques is considered from Kotsiantis (2007). The two logic-based techniques considered are *Decision Trees* and *Rule-based*, while perception-based techniques will focus on Artificial Neural Networks (ANN). *Statistical* methods that will be considered are Bayesian networks, and under instance-based techniques the *Nearest Neighbour* as stated in Kotsiantis (2007). Not all classification techniques are discussed here; this is to focus the literature on the most popular classification techniques (on which other techniques are built), for further reading on other classification techniques a breakdown of the techniques and relevant references is provided. A short summary of the different *classification* techniques, their applications, and sources where each is discussed is given in Table 2.6, as taken from the focus group held with other members of the USMA research group. Some of the sources listed are used in the discussions that follow, while others are added for completeness.

Table 2.6: Summary of classification techniques.

Classification Technique:	*Source:*	*Known for/Application:*
1 **Decision Trees**	Kim et al. (2006) Kotsiantis (2007) Larose (2014) Paramasivam et al. (2014) Rokach and Maimon (2014)	Decision support tool Customer segmentation Customer identification Fraudulent behaviour Fault diagnosis
1.1 Classification and regression trees (CART)	Breiman et al. (1984) Larose (2014) Rokach and Maimon (2014) Steynberg (2016)	Financial analysis Building predictive models Spam filtering
1.2 C4.5 algorithm	Hssina et al. (2014) Kotsiantis (2007) Larose (2014) Quinlan (2014)	Generate decision tree Text processing Fault diagnosis
1.3 Random Forest	Breiman (2001) Murphy (2012)	Fraudulent behaviour Automatic medical diagnosis Identifying stock behaviour E-commerce: Recommendation system Outlier detection

Continued on next page

	Classification Technique:	*Source:*	*Known for/Application:*
2	**Support Vector Machines (SVM)**	Coussement and den Poel (2008) Huang et al. (2007) Jansen (2007) Kotsiantis (2007) Rechenthin (2014) Tomar and Agarwal (2013) Vapnik (1999)	Text and hypertext categorisation Pattern recognition Customer segmentation Image segmentation Bioinformatics
3	**Neural Networks**	Bloom (2004) Chan (2005) Hastie et al. (2009) Izenman (2008) Jiawei et al. (2012) Kuo et al. (2006) Linoff and Berry (2011) Petroulakis and Miaoudakis (2011)	Decision making Pattern recognition Sequence recognition Face identification Automatic medical diagnosis Spam filtering Market segmentation Customer identification
4	**Naïve Bayes Network**	Jiawei et al. (2012) Rechenthin (2014) Li (2015)	Text categorisation Pattern recognition Automatic medical diagnosis Spam filtering

Continued on next page

	Classification Technique:	*Source:*	*Known for/Application:*
5	**k-Nearest Neighbour**	Kotsiantis (2007) Larose (2014) Li (2015) Rechenthin (2014) Salkind (2007)	Concept search Recommendation system Outlier detection
6	**Rule-Based Classifiers**	Cakir and Aras (2012) Ishibuchi and Yamamoto (2005) Lawrence and Wright (2001)	Automatic medical diagnosis Concept search Recommendation systems Outlier detection Loyalty programs

- Decision Trees:

  A Decision Tree (DT) is made up of nodes, branches and leaves. Each node represents a test on a feature creating a branch and after the final branching a leaf forms, which is a class label (the final outcome given the various nodes and branches chosen) (Phyu, 2009). The branch in a decision tree represents a value or the outcome of the 'test', with a decision tree having a root node from which the classification process is expanded. An example of a decision tree is given in Figure 2.14, where each node was a test, such as whether the person is a student, if so then the person is likely to purchase a computer.

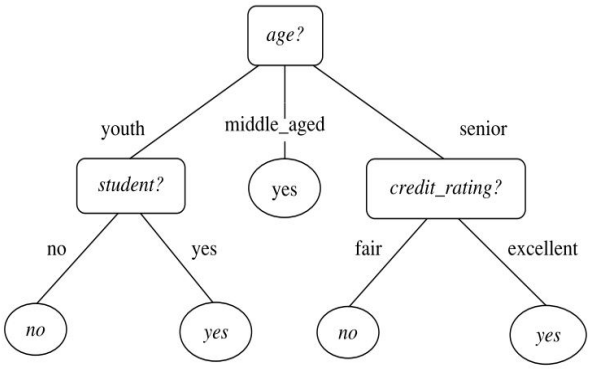


Figure 2.14: A decision tree which determines the likelihood of a person buying a computer given certain attributes (Jiawei et al., 2012).

  Most decision tree algorithms employ a top-down greedy approach during the development of a decision tree such as the CART (Classification and Regression Trees) which are binary trees and ID3 (Iterative Dichotomiser), which step through each attribute of a test sample iteratively to construct a decision tree. A common problem with decision trees is the ease at which overfitting (having a growing amount of branches) can occur. Phyu (2009) proposes two approaches to prevent overfitting: i) stopping the training algorithm when the data is perfectly fit and ii) prune the induced decision tree (remove one or more tests which yield the same or outlier/noise branch). An example of pruning is given in Figure 2.15 and in Jiawei et al. (2012)

two common pruning approaches, namely *pre-pruning* and *post-pruning* are discussed.

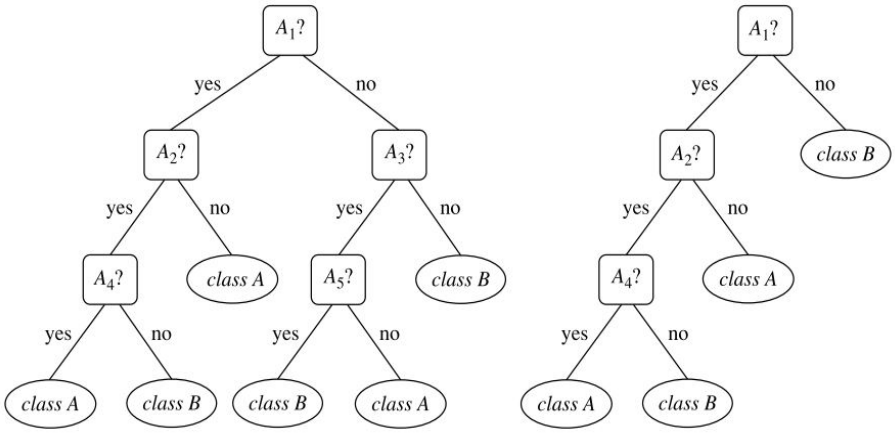


Figure 2.15: A decision tree where pruning was employed in order to remove overfitting, Jiawei et al. (2012).

The ID3 will be considered to illustrate the method by which decision trees are typically developed, as it is the method upon which others are based and is also widely used (Quinlan, 1986). The ID3 method was developed to be able to produce with a certain confidence level large decision trees with multiple attributes and objects, without being computationally expensive. The ID3 steps incrementally through each node, thereafter the developed tree is able to provide answers to all tests, the algorithm terminates. Using the notation of Quinlan (1986), let $C = \{C_1, C_2, ..., C_w\}$ be the number of objects or sample points and $O = \{O_1, O_2, ..., O_w\}$ be the number of outcomes from samples T. This is shown visually in Figure 2.16. From this, each C_i has a corresponding O_i. Using Bayes Theorem,

$$Prob(A = A_i | class = P) = \frac{Prob(A = A_i \&\& class = P)}{Prob(class = P)} = \frac{p_i}{p} \quad (2.29)$$

where there is a probability that A_i is the value for C_i given a distribution of values of A (attributes) in C, as a function of their class. This holds true, given the object belongs to class P. The test from which decision trees are built is done by branching along the attributes, therefore the root

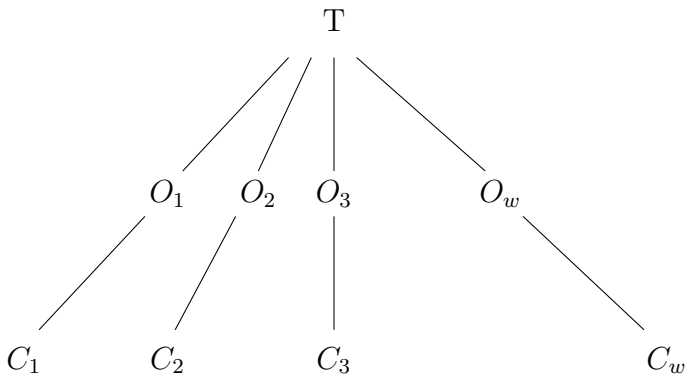


Figure 2.16: The structuring of a tree, with the objects in C (Quinlan, 1986).

of the tree is an attribute. Two assumptions are made in Quinlan (1986) for developing the decision tree, the first is that an object will belong to class P with probability $p/(p+n)$ and class N with probability $n/(p+n)$. The second being

$$I(p,n) = -\frac{p}{p+n}\log 2\frac{p}{p+n} - \frac{p}{p+n}\log 2 \tag{2.30}$$

is used by a tree to classify an object, then returning a class, P or N respectively. Given that C contains p objects of class P and n of class N. The expected information required for the sub-tree for C_i is then $I(p_i, n_i)$. The expected information required for the tree with A as the root, the weighted average is obtained using

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i). \tag{2.31}$$

The weight of the i-th branch is the proportion of the objects in C that belong to C_i. The information gained from this branching from A is

$$gain(A) = I(p,n) - E(A) \tag{2.32}$$

which is the information gained minus the expected information to be gained branching on the attribute. Using this, the largest attribute value would then be used as the root node from which to branch. This is an iterative step until the stopping condition is met, as discussed. Lastly, there is further

reading available in Quinlan (1986) where it is discussed how to develop a decision tree if there is noise or variation in the training data which can affect the outcome of the decision tree.

Using the example from Quinlan (1986) to illustrate the ID3 method, Table 2.7 is used in the analysis.

Table 2.7: A training dataset from Quinlan (1986) to demonstrate the ID3 algorithm.

No.	Outlook	Temperature	Humidity	Windy	Class
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Let C be the set of objects, therefore 9 of the 14 objects are of class P and 5 of class N. This results in a $I(p, n) = 0.94$. Considering the *outlook* attribute, five objects are *sunny*, two from class P and three from N resulting in $I(2, 3) = 0.971$. For the *overcast* outlook, the information gain is $I(4, 0) = 0$ and *rain* $I(3, 2) = 0.971$. The expected information requirement for the *outlook* attribute is $E(outlook) = 0.694$.

The gain achieved for outlook is then $gain(outlook) = 0.264$. Conducting the same process on the other attributes, the gains are, $gain(temperature) =$

0.029, $gain(humidity) = 0.151$ and $gain(windy) = 0.048$ respectively. The result of this is that the ID3 algorithm chooses the *outlook* attribute as the root from which the objects are divided and from which the tree is generated.

- Rule-Based induction:

  The basic representation of a rule is given by Oded and Lior (2005) where rules are given in a typical program-syntax format, through the use of IF-ELSE statements, whereby the rules will classify an object through testing if the object does or does not meet the conditions. There are various rule-induction algorithms, but Oded and Lior (2005) consider the Learning from Examples Module version 1 (LEM1), Learning from Examples Module version 2 (LEM2) and AQ algorithms to be the most representative.

  As stated in Witten et al. (2016), rule-based classification is a popular alternative to decision trees. The construction of a set of good rules is done by measuring and reducing the error rate measured on an independent set of data. Given in Witten et al. (2016) are two methods by which to create rule-learner algorithms, the first is a separate-and-conquer technique and the second is to create partial decision trees and from this extract rules. The LEM and AQ algorithms are separate-and-conquer algorithms.

  Similar to decision trees, rule-based induction makes use of the information gain equation, which tests the different rules together and determines the information gained qualitatively. The information gain equation is given in Witten et al. (2016) and states

$$p\left[\log\frac{p}{t} - \log\frac{P}{T}\right] \tag{2.33}$$

  where p is the number of positive instances, t the total number of instances covered by the new rule, while P and T are the number of positive and total number of instances satisfying the rule before a new test. Rules with a higher number of positive instances are then accepted. For testing the success of a rule, the ratio

$$[p + (N + n)]/T$$

can be used, where n is now the number of negative instances $n = t - p$ and N is the total number of negative instances. This success ratio is also used during reduced-error pruning operations. The problem when using this ratio is also highlighted, as the ratio treats positive and negative instances equally, which is unrealistic when there are many rules.

Following on from the logic-based techniques, perception-based techniques, specifically *Artificial Neural Networks* (ANN) are considered. A brief overview of the techniques is given along with references to adaptations of the core ANN technique and algorithms by which ANN are employed.

- Artificial Neural Networks (ANN):

  ANN are described by Basheer and Hajmeer (2000) to be structures which are made up of simple processing elements that are densely compacted together. An ANN is an approximation of the method by which the human brain conducts processing, by having electrical impulses directed through a given neuron, a 'threshold' has been exceeded. If this threshold is passed, the impulse is directed through to the next part of the brain, and a decision or action is made by a human. The working of an individual human neuron is provided in Figure 2.17, alongside the analytical ANN model based upon the human neuron and a threshold function. The given ANN model then conducts classifications according to this threshold function. Such processing elements as shown, perform large parallel computations. Further, ANN have favourable characteristics such as being able to process information which is non-linear, requires high parallelism, being able to handle imprecise information, being robust, as well as fault-and-failure tolerant. Shown in Figure 2.17, are the method by which computations are conducted by ANN, represented alongside a neuron found in humans' brains.

  An ANN operates by having processing neurons which receive inputs (stimuli) which are combined to form a net input (ξ) which must pass a threshold gate to then transmit an output y. The output will however only be transmitted once the ξ threshold has been exceeded, namely, the 'threshold limit' (b). The output is calculated by taking the dot product of the input signals

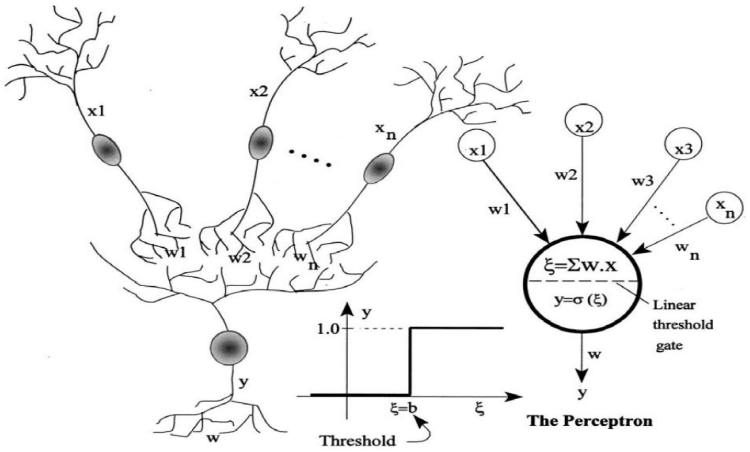


Figure 2.17: A depiction of a neuron found in human brains, and how it translated into a ANN, which comprises of various inputs and weights and an output y. Once the threshold (ξ) is reached the output is given by a binary value 1, else 0, Basheer and Hajmeer (2000).

(x) and their respective weights w (weight of connection between neuron i and j) for n signals. The output is typically given by a binary response and is given in Basheer and Hajmeer (2000) as

$$y \begin{cases} 1, & if \sum_{i=1}^{n} w_i x_i \succeq b \\ 0, & if \sum_{i=1}^{n} w_i x_i \prec b. \end{cases} \tag{2.34}$$

Further, weights greater than zero enhance the net input (ξ) while weights less than zero inhibit neuron activity. In Figure 2.17, the ANN is trained using a training sample and the weighting changes with respect to the error and output (y). The error is a function of all the weights. The threshold or activation function in the figure is given by a *binary threshold function* which is not differentiable and not favoured for back-propagation learning, Kriesel (2005). In Kriesel (2005), various 'activation' functions are given. Basheer and Hajmeer (2000) note four basic rules by which the network adjusts the weighting of neurons for further reading. The rules are the *error-correction learning* (ECL), *Boltzmann learning* (BL), *Hebbian learning* (HL) and the *competitive learning* (CL) rule. In the article, various common ANN are also given, but because of the popularity of the *back-propagation networks* algorithm, it is discussed in greater depth.

As in regression, the goal is to approximate the data with a function and reduce the error, typically the MSE (Mean Square Error function) which can also be used in ANN. This function is used in regression to reduce the error and provide accurate parameter values using functions such as the *maximum likelihood* and *least-squares* and similarly can be used in ANN. Zhang et al. (2012) derive the MSE to be (2.37), where the second term is the approximation error caused by randomness in the data. d is the dimensional inputs x, and the mapping function $F : R^d \rightarrow R^M$ and M is the vectored output y. From least squares equation, the mapping function F minimises the squared error E,

$$E[y - F(x)]^2 \tag{2.35}$$

and the conditional expectation of y given x is

$$F(x) = E[y|x], \tag{2.36}$$

from which the MSE is,

$$\begin{aligned} \text{MSE} = \sum_{j=1}^{M} \int_{R^d} [F_j(x) - P(w_j|x)]^2 f(x) \mathrm{d}x \\ + \sum_{j=1}^{M} \int_{R^d} P(w_j|x)(1 - P(w_j|x)) f(x) \mathrm{d}x. \end{aligned} \tag{2.37}$$

Applications of ANN include pattern classification which is applicable in image recognition, prediction and data analysis purposes. An example is discussed in Mao (1996), where ANN are applied to optical character recognition (OCR).

- Support Vector Machines (SVM):

  From Kotsiantis (2007) and Burges (1998), *support vector machines* (SVM) are considered to be the latest machine learning technique to be developed. SVMs separate different classes of data using a 'margin', which is a plane running through the data (Kotsiantis, 2007).

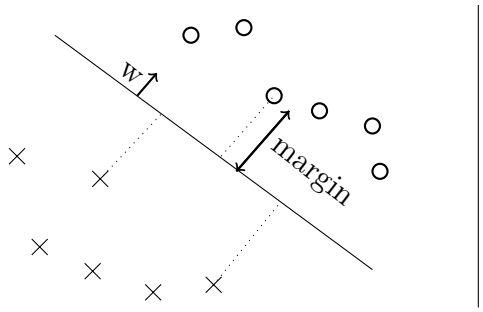


Figure 2.18: An example of a hyperplane that is created between two classes of data with a 'margin' measured from this plane using the SVM technique Tong and Chang (2001).

An example from Tong and Chang (2001) using their notation, a training dataset $\{x_1, ..., x_n\}$ of vectors and their labels $\{y_1, ..., y_n\}$ where $y_i \in \{-1, 1\}$ are given. Using this data, Figure 2.18 shows that a SVM are hyperplanes that separate the data, with one side being classified 1, and the other -1. All vectors x_i on the hyperplane are then called 'support vectors'. These training instances are extended to a *feature space* $\mathbf{F}$ to prevent non-separability that can be caused by misclassification of input data (Kotsiantis, 2007). This is done through a 'Mercer' kernel (Tong and Chang, 2001) and (Burges, 1998) such that there is a set of classifiers given as

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), \tag{2.38}$$

where $f(x) > 0$ and using K as the kernel operator. If the Mercer condition is then satisfied, the data can be rewritten into the feature space as follows, given that $K(\mathbf{u},\mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$ where $\Phi : K \to \mathbf{F}$,

$$f(x) = \mathbf{w}\Phi(x), w = \sum_{i=1} \alpha_i \Phi(x_i). \tag{2.39}$$

Tong and Chang (2001) continue by providing different kernel functions, but Kotsiantis (2007) states that the correct kernel function is dependent on the data and application. Applications of SVM include hand written digit recognition, object recognition and text classification (Tong and Chang, 2001) and (Burges, 1998).

The final techniques that are considered are the statistical techniques which are *Bayesian Networks* and *Instance-based* techniques.

- Bayesian Networks:

  A Bayesian network is a collection of *causal networks*, with the linkages between one another given by conditional probabilities (Jensen and Nielsen, 2007). A *causal network* is a collection of *causal links*. An example given is a 'car start problem, where if the car doesn't start, there are a set of possible checks that can be made, until the car is able to start again', shown in Figure 2.19. Between each link, there is then a conditional probability (probability of the one variable affecting the other) attached, linking each variable to another.

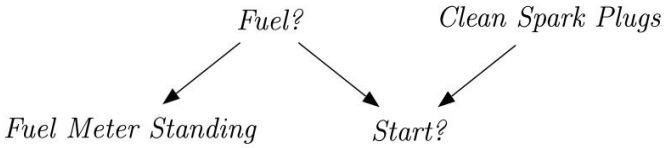


Figure 2.19: The *car start problem* used to illustrate a causal network and show each causal link, (Jensen and Nielsen, 2007).

To graphically model the probability relationships among a set of variables, a Bayesian Network (BN) can be used, (Kotsiantis, 2007). Jensen and Nielsen (2007) defines a Bayesian Network as a model that consists of the following attributes:

1. A *set of variables* and a set of directed edges *between variables*.
2. Each variable having a finite set of mutually exclusive states.
3. A DAG (Directed Acyclic Graph) is formed from these variables and directed edges. Directed graphs are only acyclic if no direct path exists from $A_1 \rightarrow A_n$ such that $A_1 = A_n$.
4. Given a variable A and having parents $B_1, ..., B_n$ a conditional probability table is associated with the variable, resulting in $P(A|B_1, ..., B_n)$.

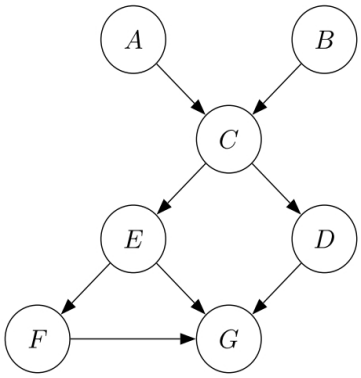


Figure 2.20: A simplified visual representation of a Bayesian Network, (Jensen and Nielsen, 2007).

The variables in a BN are given by *nodes* which are joined together, called *links* which represent causal or informational dependencies among the variables, (Pearl, 1998). Bayesian networks do not necessarily have to refer to causality, such that there is no requirement for links existing to show a causal effect. Graphically, a Bayesian Network shown in Figure 2.20 which contains several variables. Looking at variable A which has a probability of $P(A)$ associated to it, has no parent variable, as is the case with variables C, D, E, F and G. Their probabilities depend on the respective parent variables, $P(C|A, B), P(G|F, E, D)$ *etc.*

Bayesian Networks make use of the chain rule in order to create the conditional probabilities that are used to ensure the model corresponds to the real-world conditional independence properties. The chain rule, which is applicable to Bayesian Networks (BN), can be given as follows:

Let $S = \{A_1, ..., A_n\}$ which is a set of variables having a joint conditional probability given by

$$P(S) = \prod_{i=1}^{n} P(A_i|\text{pa}(A_i)). \tag{2.40}$$

$\text{pa}(A_i)$ represents the parents of A_i. Assume variables A and B are *d-separated* given variable C. *d-separation* refers to having the states each

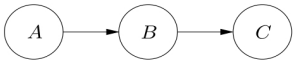


Figure 2.21: The concept of *d-separation* given a simplified node and link representation between variables A, B and C.

having a probability associated with them, A and C blocked 'communication' by knowing the certainty of B. This can be visually depicted by Figure 2.21. Jensen and Nielsen (2007) make use of the generalised chain rule and fundamental rule given by

$$P(A_1, A_2) = P(A_1|A_2)P(A_1) \tag{2.41}$$

to develop the BN chain rule, assuming variables A and B are $d-separated$ given C for example, resulting in the BN chain rule to simplify to

$$P(A_n|B, C, \text{pa}(A_n)) = P(A_n|\text{pa}(A_n)). \tag{2.42}$$

The BN chain rule allows for compact representation of a joint probability distribution.

The following from Murphy (1998), is an example of a *Bayesian Network*. All nodes have two possible values, *true* (T) and *false* (F). The Bayesian network is shown in Figure 2.22, where the variable are 'wet grass' (W) is satisfied, given that it is cloudy (C) and has rained (R), or that the sprinkler (S) is on.

The chain rule for this example can then be given as,

$$P(C, S, R, W) = P(C) * P(S|C) * P(R|C, S) * P(W|C, S, R)$$

to illustrate the joint probability distribution. Conducting probabilistic inference, Murphy (1998) then shows that if it is assumed the grass is wet, the following posterior probabilities can be calculated as follows (1 = true and 0 = false), for the sprinkler,

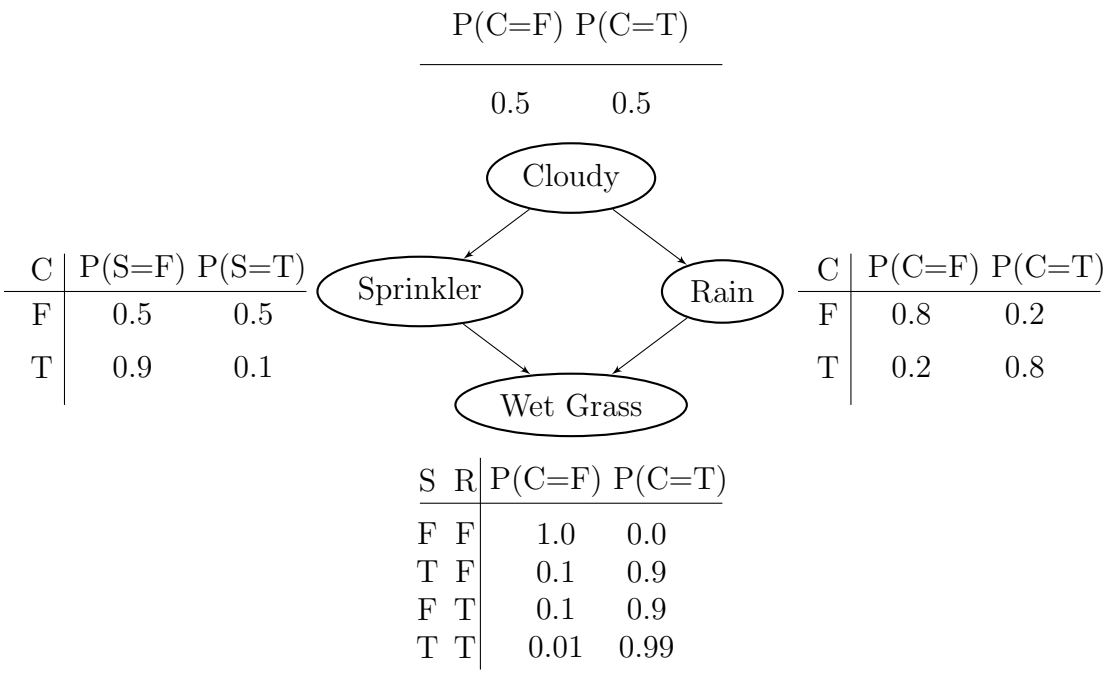


Figure 2.22: A Bayesian Network example indicating the various causes for there being wet grass and the associated probabilities for each cause summing to one, (Murphy, 1998)

$$\begin{aligned} P(S=1|W=1) &= \frac{P(S=1,W=1)}{P(W=1)} \\ &= \frac{\sum_{c,r} P(C=c,S=1,R=r,W=1)}{P(W=1)} \\ &= \frac{0.2781}{0.6471} = 0.43 \end{aligned}$$

for the rain,

$$\begin{aligned} P(R=1|W=1) &= \frac{P(R=1,W=1)}{P(W=1)} \\ &= \frac{\sum_{c,s} P(C=c,S=s,R=1,W=1)}{P(W=1)} \\ &= \frac{0.4581}{0.6471} = 0.708 \end{aligned}$$

and for the grass being wet

$$P(W = 1) = \sum_{c,r,s} P(C = c, S = s, R = r, W = 1) = 0.6471.$$

The results show that using the BN and conducting inference on the network, the results indicate that the grass is more likely to be wet because of rain. This probability can then be used for future predictions.

The final statistical technique considered is *instance-based learning*, typically algorithms used are known as *lazy-learner*, as the induction process is delayed until classification is performed (Kotsiantis, 2007).

- Instance-based learning:

  From Phyu (2009) instances are considered to be points or data points contained in an *n-dimensional* space, each dimension corresponding to a *n-feature*. Further, the absolute distance of instances is seen of less importance compared to the distance between two similarly classified instances. Phyu (2009) lists different functions that can be used to reduce the distance between similar class instances and maximise the distance between different instances.

  A disadvantage of using instance-based learning as noted in Kotsiantis (2007) is the computational time required in order to perform the classification task. Following from this, however, an advantage is the relative simplicity and accuracy of using instance-based techniques. Phyu (2009) and Kotsiantis (2007) agree that the most simplistic yet commonly used technique today is the *k-Nearest Neighbour* (k-NN) technique, which is also known as a *lazy-learner*, and will therefore be discussed further.

  Jiang et al. (2007) provide an overview of the k-NN technique which will be used in this discussion, along with Bhatia and Ashev (2010). Firstly, instances are denoted by x and are described by an attribute vector $A_i = \{a_1(x), a_2(x), ..., a_n(x)\}$ where $a_i(x)$ is the value of the i-th attribute A_i of x. C is the class variable and its respective value, $c(x)$. As with most instance-based techniques, the k-NN minimises the distance between similarly classed instances, while maximising the distance between differently

classed instances. Herein, because Jiang et al. (2007) make use of the *Euclidean* approach it will be used to measure the distances between two instances, but as stated previously, others can be found in Phyu (2009). The *Euclidean* function is given as

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(a_i(x) - a_i(y))^2} \tag{2.43}$$

where y is the other instance being compared.

Next, the k-NN equation which classifies the different instances depending on the distance is given by

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c(y_i)). \tag{2.44}$$

$y_1, y_2, ..., y_k$ represent the k number of nearest neighbours of x. The result from (2.44) is

$$\delta(c, c(y_i)) = \begin{cases} 1, & if \quad c = c(y_i) \quad \text{OR} \\ 0, & \text{otherwise.} \end{cases}$$

Jiang et al. (2007) note three shortcomings of k-NN and proposes the following: firstly, is to implement *attribute weighting* or eliminating of the irrelevant attribute. Datasets can contain irrelevant attributes known as the *curse of dimensionality* which affects the results from the *Euclidean* function. The *attribute weighting* function weighs each attribute differently (more weighting results in the attribute being 'favoured' more) and the equation is given by

$$d(x,y) = \sqrt{\sum_{i=1}^{n} {w_i}^2 (x_i(x) - a_i(y))^2} \tag{2.45}$$

with w_i from $i = 1, ..., n$ being the weight of attribute A_i. Other methods to improve the distance estimation between attributes are discussed further in the article such as the VDM (Value Difference Metric) and WAKNN (Weight Adjusted k-Nearest Neighbour).

To address the second shortcoming which is the neighbourhood size given by k, Jiang et al. (2007) propose the use of a *Selective neighbourhood Naive Bayes* (SNNB) which makes use of Bayes theorem and a Bayesian Network to automatically search through all values for the best k value. There are further algorithms discussed, DKNN (Dynamic k-Nearest Neighbour) and DKNAW as well as KDTree and NBTree (Naive Bayes Tree) which make use of decision trees to estimate the best value for k.

The final shortcoming addressed is to correctly estimate the class to which an attribute belongs. A *k-Nearest Neighbour Distance Weighted* (KNNDW) method is proposed which weighs the vote of k-nearest neighbours differently given the distance from test instance x. The KNNDW function is

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \frac{\delta, (c, c(y_i))}{d(y_i, x)^2} \tag{2.46}$$

which takes the classified instance, that is if $c = c(y_i)$, and divides it by the squared distances between two instances. If $c \neq c(y_i)$ then from (2.44) the function would divide 0 by the distance, giving zero. In the article, the KNNDW was shown to significantly outperform the traditional KNN method. There are other methods which make use of Naive Bayes found in the article for further reading. For a complete breakdown of the different Nearest Neighbour (NN) techniques, Bhatia and Ashev (2010) provide a table with the different NN techniques, advantages, disadvantages and the type of data suited to the algorithm.

Conclusion

The discussion of classification is centred around grouping the input data into different classes, typically assigning a value of 0 or 1. As shown, there are different methods, such as decision trees which divide the data until a final outcome is achieved and cannot simplify further Phyu (2009). Typical issues with decisions trees are the ease at which the data can be overfitted, but methods such as pruning can be employed (Jiawei et al., 2012) to overcome this. The method rule-based induction was also considered, specifically IF-ELSE statements (Oded and Lior, 2005), where a classification program can test and then classify the

data given (IF) a condition is met (ELSE). This is considered to be a popular alternative to decision trees (Witten et al., 2016). Next, the ANN was considered which is based upon the working of the neurons of the human brain Basheer and Hajmeer (2000). The classification by the ANN is made given multiple inputs, and if a threshold value is exceeded, the ANN classifies these inputs (Kriesel, 2005) and (Basheer and Hajmeer, 2000). Typical applications include artificial intelligence, image recognition, prediction and traditional data analysis. The support vector machines, the latest machine learning technique Kotsiantis (2007) classifies data by separating the data through the use of a 'hyperplane' and then finding the plane which maximises the distance between the plane and the different data to be classified Tong and Chang (2001). The data on one side of the plane would be given a certain classification different from that found on the other side. Applications of this method as discussed include object and text recognition, and text classification Burges (1998).

Other techniques considered were the Bayesian networks which make use of the probabilities between different variables, and classifies how each variable is connected to another (Jensen and Nielsen, 2007). An instance-based technique, the commonly used k-NN technique, was the final technique for classification considered. Similar instances are grouped together by minimising the distance between these instances by using an *Euclidean* function, and maximises the distance between differently classed instances, by doing so, classifying the input data accordingly (Jiang et al., 2007) and (Phyu, 2009).

Next, an unsupervised learning technique, namely *clustering* is discussed in further detail. This is typically being applied to data which are unlabelled.

2.3.3.3 Unsupervised learning: Clustering

Clustering is typically applied to data that has no class or label attached to it. The data therefore needs to be joined and classified according to similar groups of objects, (Witten et al., 2016). Clustering is for this reason an *unsupervised learning* technique as there are 'hidden patterns' or clusters of data that need to be identified Kogan et al. (2006) and there is only input data provided and no knowledge of the output (Alpaydin, 2010). Clustering is also seen as *density*

estimation because depending on how closely spaced the data points are to one another, different cluster groupings are estimated (Alpaydin, 2010).

Clustering as a machine learning technique has a variety of major applications that (Aggarwal and Reddy, 2014) have identified. Table 2.8 provides a list of the major applications of clustering. A short summary of the different *clustering* techniques, their applications, and sources where each is discussed is given in Table 2.9, as taken from the focus group held with other members of the USMA research group. Some of the sources listed are used in the discussions that follow, while others are added for completeness. Such an instance is the discussion of *Collaborative Filtering* which is considered a clustering technique, which has widespread usage in the field of e-commerce, providing a user with product recommendations based on how they rate other products.

Table 2.8: A collection of different applications for clustering identified by Aggarwal and Reddy (2014).

Application	**Description**
Intermediate step for data mining	Providing a compact summary of the data before applying other ML techniques
Collaborative filtering	Provides a summary of like-minded users after conducting ratings
Customer segmentation	Grouping similar customers together, but not using rating information as in the case of collaborative filtering
Data summarisation	Related to dimensionality reduction, therefore clustering creates compact data representations
Dynamic trend detection	Data that is streamed and needs to be analysed, dynamic clustering quickly identifies important patterns.
Multimedia data analysis	Taking in various media sources, video and audio alike, and clustering similar sounds or images
Biological data analysis	Providing a means of clustering and identifying significant trends in biological data such as human genome sequences
Social network analysis	Analysing the structure of a social network and identifying communities in the underlying network by means of clustering

[h]

Table 2.9: Summary of clustering techniques.

	Clustering Technique:	*Source:*	*Known for/Applications:*
1	**Clustering**	Aggarwal and Reddy (2014) Chiu and Tavella (2008) Demšar and Zupan (2013) Izenman (2008) Jacob and Ramani (2012) Jiawei et al. (2012) Kuo et al. (2006) Madhulatha (2011) Pierson and Porway (2017)	Market segmentation Product positioning New product development Selecting test markets Grouping of items Object recognition Recommendation system
1.1	Partitioning (Non-hierarchical) methods k-means k-medoids	Jansen (2007) Jiawei et al. (2012) Napoleon and Pavalakodi (2011) Rajarajeswari and Ravindran (2015)	Algorithms create single set of clusters, most effective for small/medium datasets.

Continued on next page

	Clustering Technique:	*Source:*	*Known for/Applications:*
1.2	Hierarchical methods Agglomerative (bottom-up) approach Divisive (top-down) approach	Chiu and Tavella (2008) Halkidi et al. (2001) Izenman (2008) Madhulatha (2011) Pierson and Porway (2017)	Algorithms create separate sets of clusters, each in their own hierarchical level (multiple levels).
1.3	Density-based methods DBSCAN DENCLUE	Berkhin (2002) Ester et al. (1996) Kogan et al. (2006)	The key idea is to group neighbouring objects of a dataset into clusters based on density conditions. It grows clusters either according to the density of neighbourhood objects (*e.g.*, DBSCAN) or according to a density function (*e.g.*, DENCLUE).
1.4	Grid-based methods STING CLIQUE	Andritsos (2002) Berkhin (2002) Ester et al. (1996)	These algorithms are mainly proposed for spatial data mining. Their main characteristic is that they estimate the space into a finite number of cells and then they do all operations on the quantised space.

To perform clustering, Andritsos (2002) provides a step-by-step process that can be followed:

1. Data collection: Extracting the information from relevant sources.
2. Initial screening: The pre-processing of data, cleaning, filtering, removal of noise and making use of a *data warehouse* (if required) to store the pre-processed data.
3. Representation: Similarity measures are chosen, characteristics and dimensionality of the data are examined for use in the clustering.
4. Clustering tendency: Determining if the data possesses the ability to naturally cluster or not.
5. Clustering strategy: Choosing the correct clustering algorithm dependent on the application and data.
6. Validation: Manually examining the data, making use of visual techniques to validate that correct clustering has occurred.
7. Interpretation: Combining the clustering results with relevant studies made to draw a meaningful conclusion.

Following from the list of applications shown in Table 2.8, there are different techniques used in clustering for these applications, which are considered next. Three techniques that will be discussed are *hierarchical algorithms, partitioning algorithms* (k-means), *density-based* and *grid-based algorithms*.

For this discussion of clustering, the notation $\mathbf{x} = (x_1, .., x_d)$ will be used, as researched by Jain et al. (1999) to denote a feature vector/observation of d length. For each x_i is a feature from this vector where the clustering algorithm tries to group similar x_i values into k number of clusters. As stated further, *hard clustering* (assign a definitive class) techniques make use of class labels l_i to each feature, with all classes $L = \{L_1, ..., l_n\}$ and $l_i \in \{1, ..., k\}$. *Fuzzy clustering* assigns a degree of membership for each input to a cluster (a clear class is not given for each input). A typical output from clustering is given in Figure 2.23 where the input data is combined into different groups, in this example ranging from

numbers one to seven. To compare the different algorithms used in clustering, a table can be found in Andritsos (2002) of the different clustering methods, along with the characteristics of each.

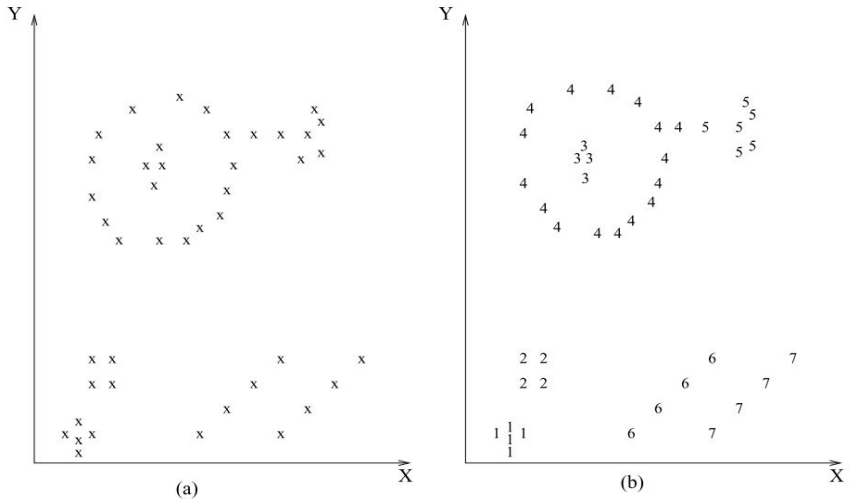


Figure 2.23: An example of the result of running a clustering algorithm, with the data being grouped into seven different classes (Jain et al., 1999).

- Hierarchical:

  The benefit of using hierarchical clustering is that there is no need for a user-defined number of clusters given as K, making it more flexible (Aggarwal and Reddy, 2014). There are two hierarchical methods, namely *agglomerative* (bottom-up) where singleton clusters are taken and more data points are added, and *divisive* (top-down) where all data points form a large cluster and the algorithm splits each point into a relevant cluster. Due to this nature of the algorithm, it forms a binary tree, where the root is representative of all the data points and the branches are the different cluster nodes. This is known as a *dendrogram*.

  From Xu (2005), using the *agglomerative* method, there are N singleton clusters and the algorithm searches for the minimum distance between clusters of class l_i and l_j

$$D(C_i, C_j) = \min_{1 \leq m,l \leq N, m \neq l} D(C_m, C_l) \tag{2.47}$$

and then updating the proximity matrix (matrix of values with the distances between points), determining the new distance between clusters. This process is repeated until all features are grouped into the same cluster. Further, there are different *agglomerative* algorithms discussed, including the *single linkage* and *complete linkage* technique.

The *divisive* algorithm works by starting with the root node (containing all the data points) and then repeatedly splits the parent nodes into two parts, C_1 and C_2 and instead of minimising the distance, maximises the *Ward's* distance, which is the opposite to (2.47). A *dendrogram* is created and then the cluster with the highest square error is chosen until singleton leaves are created (Aggarwal and Reddy, 2014). The *Ward's* criterion is as follows:

$$W(C_{a\cup b}, c_{a\cup b}) - W(C, c) = \frac{N_a N_b}{N_a + N_b} \sum_{v=1}^{M} (c_{av} - c_{bv})^2 \tag{2.48}$$

where C_a and C_b are two merged clusters, and N_a and N_b are cardinalities of clusters a and b.

This distance is weighted by a factor that is proportional to the product of the cardinalities of these two clusters (Aggarwal and Reddy, 2014).

A common disadvantage of the *hierarchical* method noted by Xu (2005) is the sensitivity towards outliers in the data, choosing the correct stopping criteria and not revisiting a cluster after it is created as stated by Kogan et al. (2006). An advantage of the method is the applicability towards any type of attribute and the flexibility with different levels of granularity (number of points).

- Partitioning: k-means

  Compared to the iterative nature of the *hierarchical* method, the partitioning method takes K number of clusters and simply assigns the data points into one of the K clusters without regard to any structure (Xu, 2005). In order to cluster the data into one of these groups, a criterion needs to be used by which to group, Xu (2005) states that a commonly used criterion function is the *sum of squared error* function which attempts to minimise

the distance between a point and the cluster centroid. The function as given by Aggarwal and Reddy (2014) as

$$\text{SSE}(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - C_k||^2 \tag{2.49}$$

and

$$C_k = \frac{\sum_{x_i \in C_k} x_i}{|c_k|}, \tag{2.50}$$

where c_k is the centroid of cluster C_k, $C = \{C_1, C_2, ..., C_k, ..., C_K\}$. The K-means algorithm works by selecting K points within the dataset X and then iteratively forming K clusters by measuring the SSE distance to the closest centroid for each data point. By iteratively recalculating the centroids for the clusters until the centroid convergence value (determined by the analyst) is met, the optimal cluster be calculated.

The advantages of this algorithm are that it allows for easy parallelisation and it does not depend on how data is organised (Kogan et al., 2006). However, there are disadvantages to the k-means method. Some of these are: the results depend on the initial guess of the centroids, the algorithm can terminate at a local optimum, which is very different from the global optimum, the algorithm is sensitive to outliers, along with others listed in Kogan et al. (2006). Aggarwal and Reddy (2014) provide a comprehensive list of other variants of the k-means algorithm depending on the application or requirements for the clustering.

- Density-based:

  Closely related to *nearest neighbour* techniques which join together two points by determining the distance a point is from one another, *density-based* techniques use this same method. This allows for clusters of arbitrary shapes to form (Berkhin, 2002) from spatial data. The density is defined in this context as the number of data points in a given volume of its locality. Density-based clusters such as the DBSCAN (Density Based Spatial Clustering of Applications with Noise) is an example of a density-based

algorithm which targets low-dimensional spatial data. Ester et al. (1996) discusses this algorithm in greater detail, and a brief overview thereof is given. The purpose of discussing this algorithm is to illustrate the use of measuring distances between points in order to form a cluster.

Each data point in a cluster must contain a given number of points within a given radius (a threshold must be exceeded). The distance function therefore determines the shape of the cluster. Two input parameters are used in the DBSCAN density-based algorithm, ϵ which is the threshold distance and *MinPts* which refers to the minimum number of points required (Ester et al., 1996).

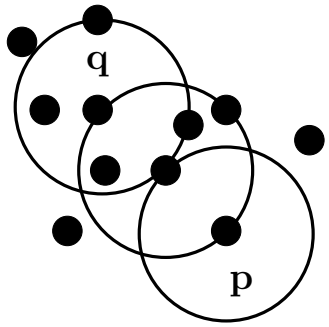


Figure 2.24: A visual representation of two points in a spatial data space. If the distance between the points p and q is less than ϵ, are they are joined into a cluster (Aggarwal and Reddy, 2014).

Using Figure 2.24 as an example, the points p and q are two data points. To determine if these points belong to the same cluster, the distance between them is determined. If the distance between the point p and q is greater than the threshold ϵ, then they do not belong to the same cluster. If there is a minimum number of points located from another *MinPts*, then a cluster has formed Aggarwal and Reddy (2014) and Ester et al. (1996). Let the distance between the two points be S_1 and S_2 then,

$$(S_1, S_2) = \min\{\text{dist}(p, q)|p \in S_1, q \in S_2\} \tag{2.51}$$

The parameters used in this algorithm need to be estimated. The literature provides detailed descriptions of the process to be followed. The advantages

of using density-based methods is the ability to develop clusters of arbitrary shapes and scalability for larger datasets (Kogan et al., 2006). The disadvantage of using density-based methods is the lack of interpretablity due to the arbitrary nature of the clusters and not being informative due to significant density changes between two adjacent areas. An example of irregular shaped clusters is given in Figure 2.25.

Figure 2.25: An example of an irregularly shaped cluster (Kogan et al., 2006).

- Grid-Based:

  Grid-based methods conduct space partitioning (Berkhin, 2002), compared to density-based methods, which determine the number of points that are grouped together to then estimate the density and partitioning. This reduces the distance between the data points and a central centroid (k-means). Further, space partitioning is based on the grid characteristics determined by the input data, being independent of the type of input data. Andritsos (2002) states that the main idea behind this method is the quantisation of the input data into cells.

  The grid-based method divides the spatial data represented in Figure 2.26 into a finite number of rectangular structures as shown. The input data would then be assigned to at first a large grid or cell, and as the algorithm progresses, the grid size decreases to cluster the data into a smaller grid/-cells. An example of such a grid-based algorithm is *STING* (STatistical INformation Grid-based) which collects a summary of statistics from the data in a hierarchical tree manner using grid cells. An example of this is given in Figure 2.27.

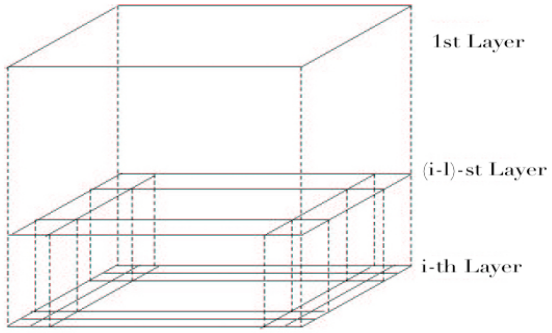


Figure 2.26: A visual representation of the rectangular grids created, wherein the data is contained (Andritsos, 2002).

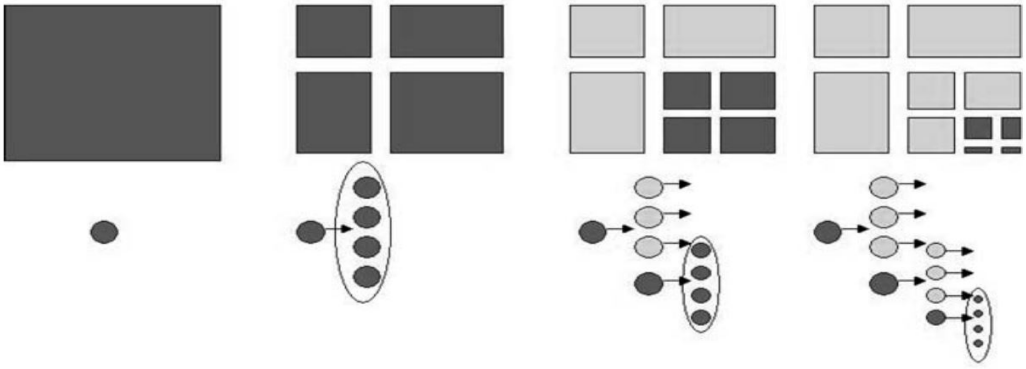


Figure 2.27: Using grid cells, the STING algorithm collects statistical information on the two-dimensional data in ever-smaller grids (Kogan et al., 2006).

After the cells are created and stored the relevant statistical information, aggregation of this data is conducted and a chi-square test is run to establish the goodness-of-fit for the cells.

- Recommender Systems: Collaborative Filtering

  Recommender systems are designed to provide a user with item recommendations, given they have not previously rated, used or seen the item before. Recommender systems are widely used in the e-commerce environment, where products or services are recommended, most famously Amazon and Netflix make use of such algorithms (Ekstrand, 2011). The goal of using such a system is to increase the sales of products and services on these sites, as users are suggested to potentially purchase products they had not

initially intended to purchase whilst browsing the site. The most widely discussed form of recommender system is known as *Collaborative Filtering*. Recommender systems can be seen as *unsupervised-learning* as there are known input variables and unknown predictor values as output, which are the recommendations made.

The first collaborative filtering method, the *user-based* technique, takes the ratings of products and services other users have provided, along with the ratings a particular user has given to other products and services and then recommends similarly rated products (Breese et al., 1998). Ratings can be of various scales, as long as the scale remains the same when recommending items on the same system (Leskovec et al., 2011). The information is typically stored as a triple (*User*, *Item*, *Rating*) which is provided to the collaborative filtering algorithm. The goal of collaborative filtering is to use these ratings to recommend products, as well as provide a 'score' of the degree to which they would likely rate or like the product or service. Another collaborative filtering method is *item-based* recommendations, where similar items are recommended to a user given other items are grouped to be similar via similarity scores (Herlocker et al., 1999). This method is more scalable and widely used in e-commerce than user-based methods. Similarly, liked and disliked items are recommended to users who share a similar preference, instead of item ratings given by specific users (Ekstrand, 2011).

An example of a recommendation problem given in Ekstrand (2011) is that of recommending movies to various users, given other users' ratings of movies. This example is shown in Table 2.10 where various users have given movies recommendations and the goal is to recommend movies of similar ratings to the users and predict what ratings the users will give the movies where no ratings are provided. This example is used further in the discussion, to demonstrate how the algorithm provides recommendations.

For this example, there are $\boldsymbol{U}$ number of users, $\boldsymbol{I}$ number of items and the ratings are given by $\boldsymbol{R}$, the ratings matrix. A rating by a specific user is then given by $r_{u,i}$ for item i. The first collaborative filtering algorithm

Table 2.10: Example of movies and user ratings, used in collaborative filtering algorithms (Ekstrand, 2011).

Users	***Batman Begins***	***Alice in Wonderland***	***Dumb and Dumber***	***Equilibrium***
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

discussed is the user-based, how the algorithm makes predictions and how the algorithm compares user-similarity.

Firstly, the similarities between users are determined by using similarity functions. These functions are then used to make predictions on how similar one user rates an item compared to another user. The first similarity function is the *Pearson Correlation* function (Ekstrand, 2011):

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{v,i} - \bar{r}_v)^2}}, \tag{2.52}$$

which takes the rating for a movie and the mean user rating and compares this to another user v, movie and mean rating to determine the correlation of the movie between two users. The next similarity function is the *Constrained Pearson Correlation* function, which takes the average rating as a 'neutral' rating r_z and in place of the mean value used in (2.52). The *Spearman rank correlation* function ranks user ratings, taking the highest rated as 1, and increasing in ranking as ratings for movies decrease. These rankings are then used in place of the ratings in (2.52). The final similarity function used in user-based collaborative filtering is the *Cosine Similarity* function. Users are represented in a I-dimensional vector space, the similarity between user ratings are the cosine distance between two rating vectors. This function is given as,

$$s(u,v) = \frac{\boldsymbol{r}_u \cdot \boldsymbol{r}_v}{\|\boldsymbol{r}_u\|^2 \|\boldsymbol{r}_v\|^2} = \frac{\sum_i r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_i r_{u,i}^2}\sqrt{\sum_i r_{v,i}^2}^2} \tag{2.53}$$

where unknown ratings are given a rating of 0, and the cosine similarity also delivers the same results as the Pearson correlation when users give the same ratings. Using these similarity functions, predictions can now be made to users. The collaborative function requires the similarity function to be of size $s : U \times U \rightarrow \mathbb{R}$ and computes a neighbourhood (N) of s-values for $\boldsymbol{U}$. The number of neighbours to select according to Leskovec et al. (2011) is domain specific, and having a smaller neighbourhood size increases accuracy (Ekstrand, 2011).

The predictions for N number of neighbours is then given by

$$p_{u,i} = \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|} \tag{2.54}$$

where user u is given a predicted rating for movie i. The user means are subtracted to compensate for users who tend to rate movies higher or lower in general. To normalise this function in order to compensate for users' rating for random movies the following additions can be made to (2.54) to give

$$p_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})/\sigma_{u'}}{\sum_{u' \in N} |s(u,u')|}. \tag{2.55}$$

Using the table and example from Ekstrand (2011), the following prediction is made for the movie *Equilibrium* for user C. The mean for the particular user is then 3.667, only two users have rated the movie, namely user A and D. Therefore the similarity values using *Pearson Correlation* function is $s(C,A) = 0.832$ and $s(C,D) = -0.515$. The predicted rating for user C for this movie, using (2.55), is then

$$\begin{aligned} p_{C,eq} &= \bar{r}_C + \frac{s(C,A)(r_{A,eq} - \bar{r}_A) + s(C,D)(r_{D,eq} - \bar{r}_D)}{|s(C,A)| + |s(C,D)|} \\ &= 3.667 + \frac{0.832 \cdot ((5-4) - 0.515) \cdot (2-3)}{0.832 + 0.515} \\ &= 4.667. \end{aligned}$$

The final collaborative filtering algorithm discussed is the item-based algorithm as from Ekstrand (2011). The item-based algorithm uses the preferences for similar items of users to make predictions and recommendations. An example is if two users similarly like two items, those two items are seen as similar. Related items are then predicted and recommended to those users. Due to this, a user's rating for an item can change if that and other users change their ratings over time.

The similarity functions used are similar to those used in user-based methods, but the *Conditional Probability* function is also used in item-based algorithms. For the item-based use case, a similarity matrix is made up of a collection of items, of size $s : I \times I \to \mathbb{R}$. The *Conditional Probability* function is commonly used in sites which relate to e-commerce, thus the explanation will be around this theme. Conditional Probabilities are based on the probabilities $s(i,j) = Pr_B[j \in B | i \in B]$, where B relates to previous purchases made by user u, item i, and item j. To prevent popular item purchases from being similar to most other items, a damping factor α is added. The resulting function is given by (2.56),

$$s(i,j) = \frac{\text{Freq}(i \wedge j)}{(\text{Freq}(i)\text{Freq}(j))^{\alpha}}. \tag{2.56}$$

To make predictions using the similarity scores for various items contained in $\boldsymbol{S}$, predictions $p_{u,i}$ can be made by using (2.57)

$$p_{u,i} = \frac{\sum_{j \epsilon S} s(i,j) r_{u,j}}{\sum_{f \epsilon S} |s(i,j)|} + b_{u,i}. \tag{2.57}$$

Using the same example as before, the similarity values were determined using *cosine similarity*, giving $s(B,E) = 0.607$, $s(A,E) = 0.174$ and $s(D,E) =$

0.382. Only using the two most similar items, the resulting prediction for user C and movie *Equilibrium* (eq) is given by

$$\begin{aligned} p_{C,eq} &= \frac{s(b,eq)\cdot r_{C,b} + s(d,eq)\cdot r_{C,d}}{|s(b,eq)| + |s(d,eq)|} \\ &= \frac{0.607 \times 5 + 0.832 \times 2}{0.607 + 0.832} \\ &= 3.84 \end{aligned}$$

Using these methods, recommendations can be made. There are, however, other algorithms discussed in Ekstrand (2011) and Su and Khoshgoftaar (2009), for further reading. To evaluate the algorithms, the *Mean Absolute Error* (MAE) can be used (Ekstrand, 2011):

$$\text{MAE} = \frac{1}{n}\sum_{u,i} |p_{u,i} - r_{u,i}|$$

The MAE is largely used for static value evaluations, for the predictions. When the recommendations are evaluated over time (*e.g.* performance over time as more users make use of a service), the Time Averaged Root Mean Square Error (TA-RMSE) can be used (Ekstrand, 2011).

$$\text{TA-RMSE} = \sqrt{\frac{1}{n}\sum_{p_{u,i:t_{u,i}\leq t}} (p_{u,i} - r_{u,i})^2} \tag{2.58}$$

When choosing the appropriate algorithm, the following need to be accounted for, the *target domain*, the *context of use*, *computational performance required* and finally, the *user needs* (Su and Khoshgoftaar, 2009).

Conclusion

Unsupervised learning is comprised of the different clustering techniques available. Clustering is used when the data that is given as input has no label (Witten et al., 2016). The clustering techniques discussed take this unlabelled data and in the case of *hierarchical* clustering, employ *agglomerative* or *divisive* methods to develop clusters. The k-means partitioning method determined the minimum

distance of each data point from a centroid, and iteratively moved the centroid to the position where the distances were at a minimum. The density-based method determined the distance between two points, given that the two points are within a certain distance and there are a minimum number of points a cluster is then formed (Ester et al., 1996). The grid-based technique successively reduced the grid or cell sizes to cluster nearby data points together. The final technique considered was *collaborative filtering*, which is commonly used in e-commerce to recommend products to customers based on previous ratings provided for different products by other customers, known as *product* or *rating-based* recommender. The other collaborative filtering technique was *item-based*, where items are recommended based on a user's preference towards that or other items.

The final machine learning tool considered is *reinforcement* learning.

2.3.3.4 Reinforcement learning

When making predictions or grouping unknown data together, *supervised* and *unsupervised* learning techniques are preferred learning techniques. But when the user requires a system to learn from its surrounding environment, *reinforcement* learning is used. Examples from Dietterich (1997) are the system developed which learned to play board games, and in robotics where a robot is required to map out its nearby surroundings. As stated in section 2.3.3, the goal of *reinforcement* learning algorithms is to determine the optimal policy (set of actions to execute) π^* which will maximise the cumulative discounted reward (Alpaydin, 2010).

The reward function used in *reinforcement* learning is given by Dietterich (1997) as $R(s_t, a, s_{t+1})$ where s_t is any given state at time t, a is an available action from a collection $A = \{a_1, a_2, ..., a_n\}$ and s_{t+1} is the new state after executing action a. The *cumulative discounted reward* from Dietterich (1997) and Barto and Mahadevan (2003), using this reward function is

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a, s_{t+1}) \tag{2.59}$$

where $0 \leq \gamma < 1$ is a discount factor that determines the importance of a short-term and long-term reward. This is a collection of the rewards provided to the system given that the correct policies or actions were taken. To further this reward

function, a probability of obtaining this reward is attached to the likelihood of executing the correct policy to yield the reward, known as the value function of policy π^*, given by

$$f^{\pi}(s) = \sum_{s} P(s'|s, \pi(s)) \cdot [R(s, \pi(s), s') + \gamma f^{\pi}(s')], \tag{2.60}$$

where the probability of state s' is reached given action $\pi(s)$ is taken in state s, $P(s'|s, \pi(s))$ (Dietterich, 1997).

This is known in literature as a *Markov Decision Process* because of how the environment interacts or behaves with an agent or system, given its actions (Maia, 2009). Using classification techniques discussed in section 2.3.3.2, the decisions can then be classified so that the policy (set of decisions) maximise the value function (2.60), hence the reward given.

Next, a method is discussed which is used in data analysis and before or after machine learning to improve the rate at which an algorithm performs, or visualises a feature-set, by reducing the number of features, namely Dimensionality Reduction and the Principal Component Analysis technique.

2.3.4 Dimensionality reduction

When conducting a Big Data Analysis, there can be a large number of variables contained in the dataset being analysed. Video, sound, and multi-feature datasets contain large numbers of dimensions that need to be reduced (Van Der Maaten et al., 2009). The goal of performing dimensionality reduction is to take this input data and reduce the number of features so that enough information is contained in this condensed dataset to describe the original dataset (Williamson et al., 2015). Having a dataset that has a reduced number of features provides an increase in the rate at which machine learning algorithms can perform an analysis, as well as providing a means to visualise the dataset, if reduced to a small enough feature-set to visualise (two-dimensional and three-dimensional datasets). There are different dimensionality reduction methods available, from *linear* to *non-linear* methods, Van Der Maaten et al. (2009) provides a taxonomy of all the different techniques. For this section however, the most widely discussed method in literature, namely

Principal Component Analysis (PCA) will be presented, which is a *linear* method. This is to focus the literature and provide a brief overview of the concept.

The goal of PCA is to take the input data and project the data to lower dimensions known as *principal components* (PCs). The number of principal components must be fewer than the number of features or number of datapoints in the dataset Lever et al. (2017) and Van Der Maaten et al. (2009). The PCA step conducts the projection of the data to a lower dimension in a linear fashion, similar to linear regression, but minimises the perpendicular distance between datapoints and PCs, instead of the response variable and predicted value (as in linear regression). PCA takes the input data and projects it onto the first PC, minimising the distance between the data and the PC, this in turn maximises the variance (σ^2) of these points. This process is conducted on each specified number of PCs. Each PC is also geometrically orthogonal from another (each PC has no correlation with another).

The following is an overview of the working of PCA, given by Jolliffe (2005), Smith (2002) and Shlens (2014). Suppose there is a dataset $\mathbf{X}$ matrix of size $m \times n$, with m the number of rows and n the number of columns. The matrix is to be transformed to Y using a matrix $\mathbf{P}$ of size $m \times m$ ($\mathbf{Y} = \mathbf{PX}$). The rows of P are vectors $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, ..., \mathbf{p_n}$ and the columns of X are column vectors $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_n}$ to give,

$$\begin{aligned} \boldsymbol{PX} &= (\boldsymbol{P}\mathbf{x_1} \quad \boldsymbol{P}\mathbf{x_2} \quad ... \quad \boldsymbol{P}\mathbf{x_n}) \\ &= \begin{pmatrix} \mathbf{p_1} \cdot \mathbf{x_1} & \mathbf{p_1} \cdot \mathbf{x_2} & ... & \mathbf{p_1} \cdot \mathbf{x_n} \\ \mathbf{p_2} \cdot \mathbf{x_1} & \mathbf{p_2} \cdot \mathbf{x_2} & ... & \mathbf{p_2} \cdot \mathbf{x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p_m} \cdot \mathbf{x_1} & \mathbf{p_m} \cdot \mathbf{x_2} & ... & \mathbf{p_m} \cdot \mathbf{x_n} \end{pmatrix} \\ &= \boldsymbol{Y}. \end{aligned} \tag{2.61}$$

The equation shows the projection of $\boldsymbol{X}$ on the columns $\boldsymbol{P}$. These rows of $\boldsymbol{P}$ onto which the values are projected will become the principal component directions. The next step is establishing the directions where the variance is maximised, using

$$\sigma^2 = E[(Z - \mu)^2]$$

where Z is the random variable and μ is the mean. Next of interest is the covariance (how two variables change with respect to each other), the goal is for the covariance of variable to be as close to zero, therefore creating variables being as uncorrelated as possible in a matrix $\boldsymbol{C_Y}$. The covariance matrix that is generated is given by

$$\begin{aligned}\boldsymbol{C_Y} &= \frac{1}{n-1}\boldsymbol{Y}\boldsymbol{Y}^T \\ &= \frac{1}{n-1}(\mathbf{PX})(\mathbf{PX})^T \\ &= \frac{1}{n-1}(\mathbf{PX})(\mathbf{X}^T\mathbf{P}^T) \\ &= \frac{1}{n-1}\mathbf{P}(\mathbf{X}\mathbf{X}^T)\mathbf{P}^T \end{aligned} \tag{2.62}$$

with further derivations of the covariances formula, for each component in the literature. (2.62) represents the covariance matrix function already combining the desire to transform $\boldsymbol{X}$ and $\boldsymbol{P}$ in terms of $\boldsymbol{Y}$. The matrix $\boldsymbol{P}$ is chosen such that the covariance matrix is orthogonal, which in turn maximises the variance whilst minimising the covariance between variables. This covariance matrix contains the principal components (rows of $\boldsymbol{P}$) contained in matrix $\mathbf{X}\mathbf{X}^T$, which are the eigenvectors of this matrix. This calculation also provides a means of identifying the components with the largest amount of variance contained, with the first column containing the largest amount of variance and the amount of variance decreasing in the second and, third row, *etc.*

The following example is provided in order to demonstrate this technique of dimensionality reduction. The example uses the *Iris-dataset* commonly used for machine learning development and testing. The dataset contains 150 datapoints with four features which are indicative of the specific species of iris plant.

Table 2.11: The covariance matrix of the Iris dataset.

Feature 1	**Feature 2**	**Feature 3**	**Feature 4**
0.6800	−0.0306	1.2673	0.5307
−0.0306	0.1810	−0.3136	−0.1315
1.2673	−0.3136	3.1026	1.2781
0.5307	−0.1315	1.2781	0.5868

Table 2.11 contains the values of the covariance matrix $\boldsymbol{C_Y}$ for the Iris dataset. The length of the matrix corresponds with the number of features. Using this matrix, the principal components of the dataset can be determined. For this analysis, the dataset is reduced from four features to two principal components. Using this information, the variance of the two principal components can be calculated, and how much variance each component holds is given in Table 2.12. The table shows that 92% of the variance of the dataset is contained within the first component, and as expected, the amount of variance in the following component is less.

Table 2.12: Table showing the fraction of variance that is contained within each principal component. The first component contains most of the variance of the dataset.

Principal Component 1	**Principal Component 2**
0.9239	0.0534

The component values (150 in total) of which only 12 are shown, are provided in Table 2.13. Each value in the table is a condensed value representing the four feature values, in a lower dimension. Because of this, using the covariance matrix, the values in Table 2.13 can be restored to the original dataset. An analyst can then use this information to plot the results or use the compressed data in a further analysis using machine learning tools.

Table 2.13: The Principal Component values for the Iris dataset.

Principal Component 1	**Principal Component 2**
−2.7336	−0.1633
−2.9080	−0.1308
−2.7649	−0.3048
−2.7461	0.3403
−2.2968	0.7535
−2.8391	−0.0756
−2.6442	0.1766
−2.9068	−0.5642
−2.6912	−0.1005
−2.5236	0.6579
−2.6311	0.02771
−2.8058	−0.2214

In the next section, software available for Big Data Analysis, *Hadoop* is discussed. This software can be used to store and analyse data in a parallel, scalable and fault-tolerant fashion.

2.4 Hadoop

To ensure fast and reliable processing of ever-growing large datasets, a system is required which can store and process data, whilst making use of current technologies. This lead to the development of *Hadoop*. Hadoop, also known as *Apache Hadoop* is a system developed by the *Apache Software Foundation* (ASF), to provide a scalable, distributed parallel computing capability for large datasets (Foundation, 2014). The current technological limits of hard disk read and write speeds of around 100MB/s create a bottleneck when conducting a large scale analysis (White, 2012). To overcome this, Hadoop takes large datasets, then breaks it up into smaller *blocks*, which are distributed across multiple storage disks. The Hadoop system then collects the data from multiple storage nodes, reducing the time required to read and analyse all the data due to the parallel configuration.

Looking at the hardware, the system makes use of *nodes*, which are processing and storage elements largely comprised of a CPU (Central Processing Unit), and

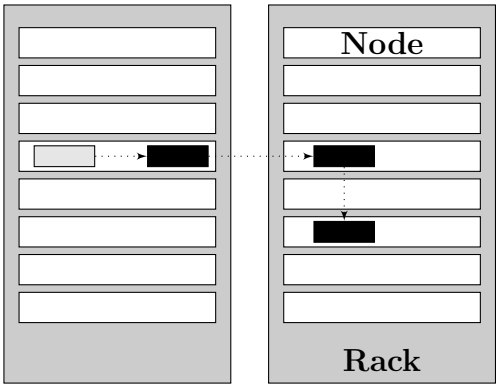


Figure 2.28: Shown is an example of multiple nodes joined together to form a rack and how data is stored and flows across multiple nodes and racks (White, 2012).

hard-disk drives. Nodes are connected together to form a *rack*. Data centres used in industry can comprise of multiple racks and nodes which work in parallel to conduct an analysis. An example of a *rack* and *node* system is given in Figure 2.28, where data is distributed across multiple nodes and racks (White, 2012).

Traditional databases are known as Relational Database Management Systems (RDBMS) store and manage only structured datasets. Standard Query Language (SQL) is the most widely implemented RDBMS which partitions databases into tables that are grouped together by subject area. Data is therefore divided into various entities (tables) that are linked together (Tekiner and Keane, 2013). The distinguishing factor between Hadoop projects and traditional databases is the ability of Hadoop to store any data type on inexpensive hardware that is distributed, making it fault-tolerant, scalable and able to process data rapidly (Bakshi, 2012).

Due to the ability of Hadoop to store data of any type is favoured for applications which make use of structured, semi-structured and unstructured data (Das and Kumar, 2013). RDBMSs such as SQL, however, are widely used in industry where structured data is used, due to its ability to easily communicate industry data in an understandable format, and when data sources are largely made up of structured data (Tekiner and Keane, 2013).

Hadoop comprises of different projects also known as modules, developed to meet various needs and applications found in Big Data, from providing databases which are distributed but structured, tools which provide management and monitoring capabilities to providing a library of processing tools and programs. Table 2.14 outlines the different Hadoop projects from Foundation (2014), each designed for specific Big Data applications.

As seen in Table 2.14, there are many Hadoop projects that have been developed. To focus the literature, the Hadoop Distributed File System *HDFS* and *Hadoop MapReduce* projects are discussed in further detail, as Tekiner and Keane (2013), White (2012), Bakshi (2012), Dittrich and Quian (2012) and Dhote et al. (2015) have concluded that these are the major components which have driven the adoption of Big Data, and allowed for High Performance Computing (HPC) to be available to non-traditional users.

Firstly, the *HDFS*, which provides the means to store large volumes of data is considered, then the *MapReduce* process is discussed, which provides the means to analyse and process large volumes of data within a short period of time.

2.4.1 Method for storing data in Hadoop: HDFS

As mentioned, Hadoop makes use of *nodes* to store incoming data of any type. The data is then duplicated across multiple nodes and racks as shown in Figure 2.28 (Verma et al., 2015). The HDFS makes use of a write-once-read-many model (Hanson, 2011) and duplicates the data across the nodes.

The *HDFS* makes use of a *master* and *slave* node system, where the *master* node, known as the *NameNode*, stores metadata of all storage nodes (nodes which store the data) and controls how data is distributed across all storage nodes (Dwivedi and Dubey, 2014). The *slave* nodes, known as *DataNodes*, are responsible for the storage of data. These *slave* nodes execute the work of the master node (Patel et al., 2012) after a query is submitted to the cluster. Figure 2.29 illustrates the process followed to store data in the HDFS format.

A user assigns a unique name for the HDFS file which serves as an identifier to all the data stored and distributed under the unique name (the name under which all metadata is stored). The data to be stored is then divided into *blocks*

Table 2.14: The various Apache Hadoop projects, each designed for specific applications in the Big Data environment (Foundation, 2014).

Hadoop Project	**Description**
Hadoop Common	A collection of the regularly used tools in other Hadoop projects/-modules
HDFS	A distributed file system designed to provide high-throughput of application data, be fault tolerant and scalable
Hadoop YARN	A Hadoop project designed for job scheduling and cluster resource management applications
Hadoop MapReduce	A system developed to process large datasets in parallel based on the YARN framework
Ambari	A web-based tool providing monitoring and managing for *HDFS, MapReduce, Hive, HCatalog, HBase, Zookeeper, Oozie, Pig* and *Sqoop* support
Avro	A Hadoop project which provides a system to serialise data
Cassandra	A project which provides a database with no points of failure and scalability
Hbase	Provides a distributed scalable database, applicable for structured data (large tables)
Hive	An infrastructure providing a user with summarisation and ad hoc querying for data warehouses
Mahout	A library for Hadoop which provides scalable machine learning and data mining tools
Pig	A data-flow language and execution framework for parallel computations
Spark	Computation engine which provides a support for various applications from machine learning, stream processing and graph computations
Tez	A general framework built atop *YARN* providing a powerful and flexible engine for batch and stream processing. Tez is being adopted in Hive and Pig as replacement for the current processing engine, *MapReduce*
Zookeeper	A coordination service for distributed applications while providing high performance
Chukwa	A project developed to manage data collection for large distributed systems

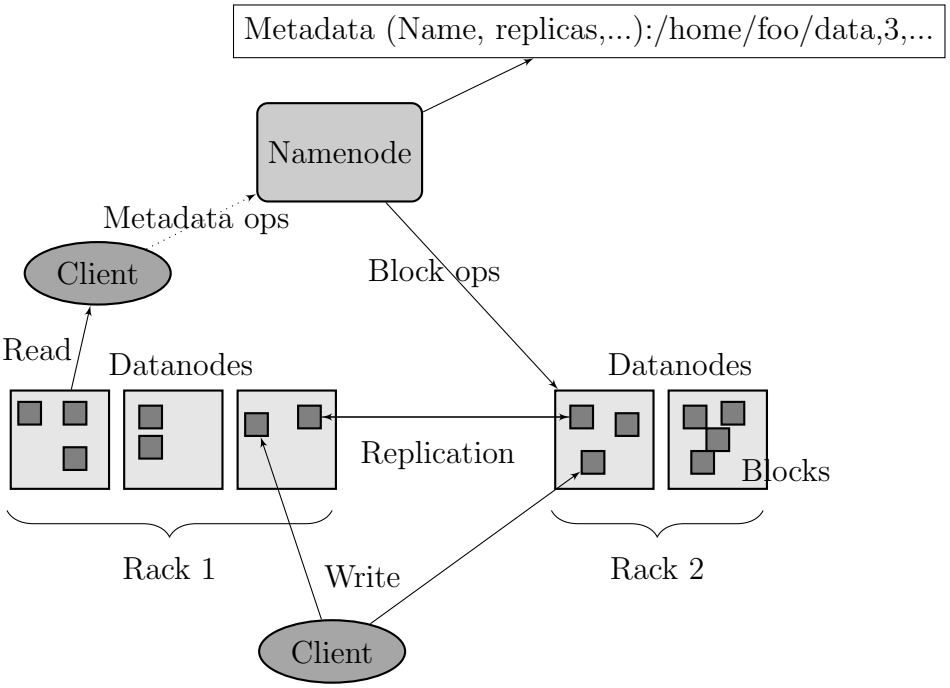


Figure 2.29: The process by which data is stored and distributed across multiple nodes in HDFS (Foundation, 2014).

and these individual blocks are then replicated, distributed and stored across multiple *DataNodes* (Ghazi and Gangodkar, 2015). Currently *blocks* are divided into a maximum of 64 or 128MB due to current technological limitations, but will in future increase in size (White, 2012). A *SecondaryNameNode* is also used to monitor changes made to the *NameNode*. The changes are then logged and updated to ensure the processing rate is not affected by any possible changes to the specific *HDFS* file created (White, 2012).

2.4.2 Method for processing data in Hadoop: MapReduce

Developed by Google, the programming model was developed to process large amounts of datasets in parallel, utilising the *Google File System* (GFS) (Hashem et al., 2016). Just like the *HDFS*, *MapReduce* is designed to be scalable and fault-tolerant and makes use of computing *nodes* Lee et al. (2011).

MapReduce can be regarded as a two-phase process, namely a *Map* and *Re-*

duce process (Dittrich and Quian, 2012). During each phase, a dataset is made of multiple key-value pairs, which are randomly distributed in the input, and thereafter a resulting key-value output (Condie et al., 2010) after computations were conducted and the distributed key-value pairs were joined. The *key* refers to a unique key or identifier for the specific data and the *value* is the specific data value linked to that key, the input being $(k1, v1)$. Similar data values can have the same key (Lee et al., 2011). There can therefore be a key k1 with multiple similar values $(v1, v2, ..., vn)$ Hashem et al. (2016) which come from intermediate keys-value pairs $(k2, v2), (k3, v3), ..., (k_n, v_n)$. The remaining keys $k2, k3, ...k_n$ refer to other data and data values associated therewith. The different keys and values can be seen in Figure 2.30.

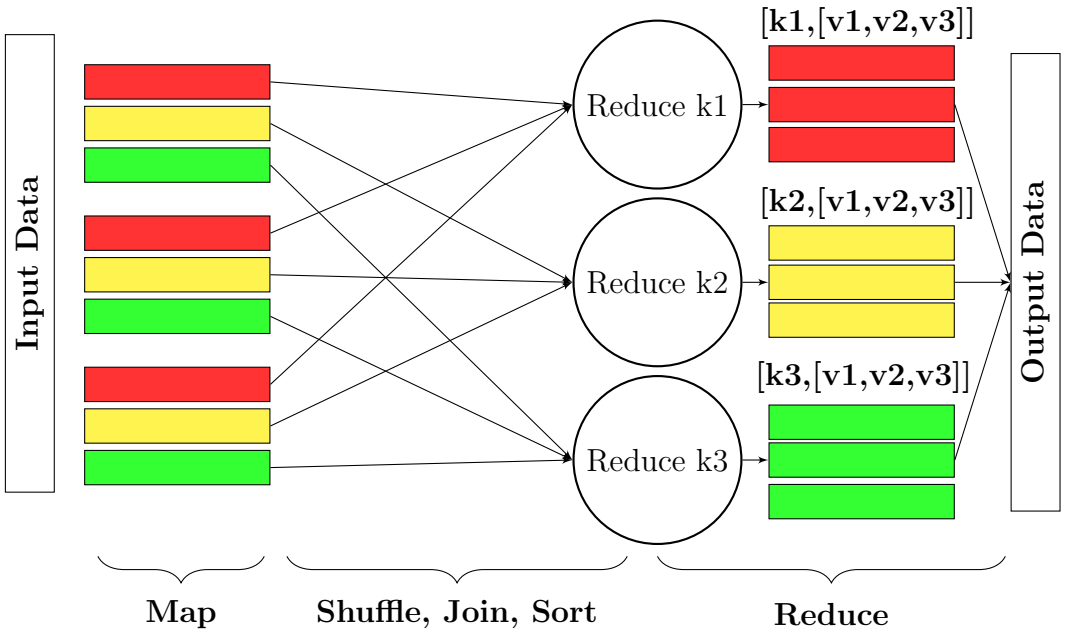


Figure 2.30: The Map and Reduce process using the key-value pair notation. Each colour refers to a key, with three values.

Using Figure 2.30 as a visual guide, during the *Map* phase, the key-value pair, $(k1, v1)$ and the $k1$ being the desired key are inserted/input (user defined), and intermediate key-value pairs are generated (mapping) $(k2, v2)$,$(k3, v3)$,...,(k_n, v_n) of the data. Similar key-values are then joined and sorted as shown in Figure 2.30 (Ghazi and Gangodkar, 2015). From the figure, all values associated with

the key ($k1$) are then joined and sorted during the *Reduce phase* Hashem et al. (2016), to give $(k1, v1, v2, v3, ...)$ as output.

Next, the *Hadoop MapReduce* framework of *MapReduce* is considered. This is a system *MapReduce* framework developed within the *Hadoop* environment to allow for large-scale parallel computations (Lee et al., 2011). Just as with the HDFS, Hadoop MapReduce makes use of a master/slave node system in order to conduct the computations in parallel (Dwivedi and Dubey, 2014). The master node used in *Hadoop MapReduce* is the *JobTracker* and the slave nodes are *TaskTrackers* (Hashem et al., 2016) and are known as a *MapReduce engine*. Figure 2.31 indicates the working of MapReduce within the different nodes, this interaction between the nodes can be seen in Figure 2.31. and how the MapReduce nodes relate to those used by the HDFS, namely the data exchange between the *NameNode* and *JobTracker*.

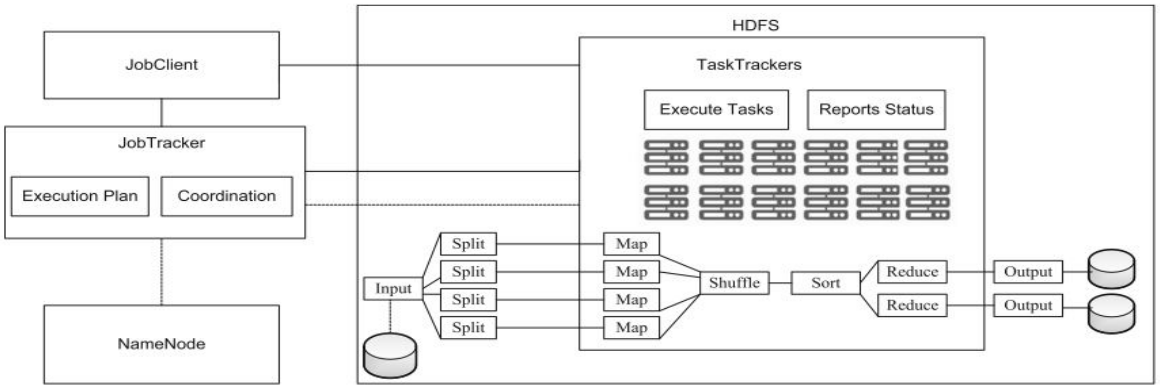


Figure 2.31: The Map and Reduce process executed within the *Hadoop MapReduce* environment (Hashem et al., 2016).

The function of the *JobTracker* node is to monitor the *MapReduce* process and it is also where an analyst or user submits queries/jobs to perform certain *MapReduce* tasks (White, 2012). The *JobTracker* node then collects the necessary data from the *NameNode* used in HDFS and locates the appropriate *TaskTrackers*. The jobs are sent to the *TaskTrackers*, which perform the parallel computations. During the computational process, the *TaskTracker* sends 'heartbeat' messages to the *JobTracker* node to ensure the task is being executed as required. The *Task-*

Tracker is the node where the *Map* and *Reduce* processes are executed (Ghazi and Gangodkar, 2015).

2.5 Spark

Following on from Hadoop in section 2.4, where the concepts of the HDFS-format and MapReduce were introduced, in this section, a new framework is discussed, namely Spark. Spark was developed in the UC Berkley RAD Lab in 2009 and was made open-source in 2010 (Ohlhorst, 2015). It is built atop of *Mesos*, a cluster operating system (allowing for parallel applications to share resources) (Zaharia et al., 2010). As stated in Cloudera (2017), Spark is considered to be a successor to the MapReduce framework to conduct data processing. The Spark framework is stated to also be one hundred times faster than similar MapReduce tasks (Spark, 2018).

The Spark framework is designed to use in-memory clusters (conduct processing using memory) instead of reading and writing to disk, enabling faster processing (Zaharia et al., 2012). The Spark framework was developed out of the need to provide a faster method to analyse data using non-trivial machine learning algorithms than what are provided by MapReduce. Multi-pass algorithms (iterative algorithms) and interactive data queries (used to explore the data) required multiple read-and-write to the HDFS (on-disk) steps, which increased the time to process data (Zaharia et al., 2012). The solution provided by Spark is the development of *Resilient Distributed Datasets* (RDD) which provides the fault-tolerance required by Big Data and in-memory storage. Further, Spark allows for applications to be developed with less effort, as processing jobs do not need to be written as *Map* and then *Reduce* tasks. Spark also provides various APIs (Application Programming Interfaces) to allow for programs to be developed in multiple different languages with less effort. As of writing this, Spark offers the ability to develop Spark applications in *Scala*, *Python*, *Java*, and *R* without the need to convert code into Java. To similarly use these or other languages other than Java, streaming and other methods to convert code to Java would be required when working with MapReduce (Spark, 2018).

Some other advantages of Spark are the addition of built in libraries for machine learning (MLib), streaming (conducting analysis in near real-time for on-line applications) called *DStream*, graphing (*GraphX* and SQL queries (*Dataframes, Datasets*). A full list of the various functions provided by each of the libraries within Spark is provided by Spark (2018). Spark as mentioned is built to be fault-tolerant due to the design of tracking each transformation undergone by a RDD (Ankam and Guller, 2015). The concept of RDD and how it ensures fault-tolerance is given by Zaharia et al. (2012): "Each RDD tracks the graph of transformations that was used to build it, called its lineage graph, and reruns these operations on base data to reconstruct any lost partition".

A summary of the tasks appropriate for MapReduce and Spark is provided by Bekker (2017) shown in Table 2.15.

The concepts of the RDD and how Spark manages and distributes the data and processing are discussed next.

2.5.1 Method for processing data: Spark

The *Resilient Distributed Datasets* (RDD) concept that Spark uses was developed to overcome problems associated with *MapReduce* as discussed in section 2.5. MapReduce runs all tasks by reading and writing to disk, which creates a lot of 'overhead' as stated by Zaharia et al. (2012) when working with complex machine learning algorithms. Such algorithms rapidly increase the time to conduct an analysis due to this 'overhead'. This is because each time a part of data is transformed (action is conducted on the data), a reading and writing process needs to be executed in MapReduce; instead Spark runs all processes in memory.

A RDD in each node as stated by Zaharia et al. (2010), RDDs can be constructed in four manners. The first is loading a file, from a shared file system such as HDFS, loading a previously generated array developed in the Scala programming language, transforming existing RDDs being used in the application or persisting a RDD. By persisting a RDD, the information is either cached (stored) to be used by another application, or writing the RDD to disk to be stored in a file or file system such as HDFS. Another property of RDDs is that they are

Table 2.15: The various Apache Hadoop projects, each designed for specific applications in the Big Data environment, (Bekker, 2017).

Tasks Appropriate for MapReduce	**Tasks appropriate for Spark**
Linear processing of data: For large parallel processing of data which do not involve non-trivial processing	Fast data processing is required: By using in-memory processing (RAM)
	Iterative processing: When there is a requirement to process data repeatedly, RDD are more appropriate, allowing multiple mapping operations (transformations)
	Near real-time processing: In applications where immediate insights and results from the data are required
Economical solution, given no immediate results required: If the rate at which processing is conducted is of less importance	Machine learning: Making use of machine learning algorithms to analyse data, spark provides a built-in API MLib. Hadoop requires third-party API's
	Joining datasets: More appropriate when it is required to quickly merge different datasets, but MapReduce is able to handle larger volume data manipulations at a slower rate

immutable, which means they cannot be modified. To make changes, transformations need to be conducted (which in turn ensures the fault tolerance) on a RDD.

The RDD, which is a collection of split data packets (partitions), are distributed across multiple nodes, is responsible for the processing of the information. This process is managed by a *JobTracker* on a *Spark driver*, and processing of RDDs is done by *TastTrackers* on *Spark executors*. This is similar to Hadoop, where the JobTracker which is located on Spark drivers, manages the distribution of the RDD and stores Metadata of the RDD. The worker node is where data is stored to be used in the analysis by a TaskTracker. A visual representation of this process is shown in Figure 2.32, from Cloudera (2017). A job is sent from the

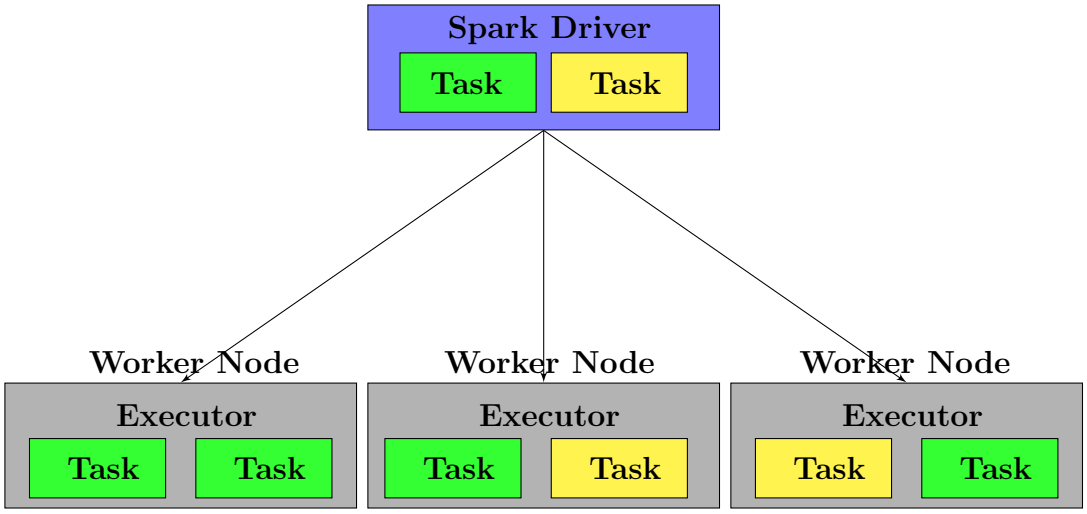


Figure 2.32: The analysis process conducted when using RDDs in Spark, The Spark driver manages and sends partitions of the RDD by the JobTracker to be stored on executors, which complete a specified task in the TaskTrackers. The RDD is partitioned on all executors, creating a RDD which is fault tolerant Cloudera (2017) and Ankam and Guller (2015).

driver to executors located on worker nodes, where the RDD is partitioned across the nodes. The processing is then conducted by the TaskTracker and results sent back to the driver.

To make use of Spark, Cloudera (2017) provides examples using the interactive Spark Shell (available for Python and Scala), shown in Figure 2.33. A shell is an environment where applications can be created and run. From this, Spark jobs can be submitted to query data. Because spark offers APIs for multiple languages, Spark can also be used in any IDE (Integrated Development Environment) if an analyst chooses not to work in the available shells for program development. Spark libraries need to simply be added.

The discussion that follows provides context around the Semantic Web and RDFs. These components were fundamental in providing contextual information, drawing from sources of Big Data, when making use of the internet.

```
$ SPARK_HOME/bin/pyspark
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information
...
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version ...
      /_/

Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

Figure 2.33: The Spark shell that is provided in Spark to conduct analysis, the shell is configured for Python (Cloudera, 2017).

2.6 Semantic web and resource description framework in Big Data

In this section, the role that the *semantic web* and *Resource Description Frameworks* (RDF) have in Big Data are discussed. The concept of *ontologies* which are used in semantic web design, are briefly reviewed. After this, the RDF which relates to the semantic web is discussed.

The *semantic web* and RDF are not directly involved or applicable in the work that is to be conducted for this project, but these concepts are big components in the Big Data environment.

2.6.1 Ontologies

The definition and understanding of an *ontology* follows that of Staab and Struder (2009), of which there are two types of ontologies defined. The first is ontologies from a philosophical discipline, which deals with the nature and structure of reality. The second, and what will be considered throughout this section is that a computational ontology (used in computer science) is a means to formally

model the structure of a system, namely different entities (data, information or words) and the interrelationships between them which are machine processable (Sicilia, 2014). Noy (2004) provides a survey of various ontology-based methods of integration on the semantic web, an example is the *prompt* machine learning system which supports ontology merging of different data entities. The end result allows for queries be to answered and data transformations to be made.

2.6.2 Semantic web

According to Allemang and Hendler (2011), the semantic web is designed to support a distributed web at a data-level rather than at the presentation level (one web page pointing to another). The end result is that data is given meaning such that machines can understand the relationship between different sources (context) (Mika, 2004). This is done through URIs (Uniform Resource Identifiers) (Berners-Lee et al., 2001). This is different from a URL (Uniform Resource Locator) which is used in current Web infrastructure to link (identify) web pages. The data from this page is however not linked with other data points (information or phrases) from other pages which prevents information regarding a single entity being shared throughout the entire web. This creates voids of information on other pages or makes it difficult to locate information. Allemang and Hendler (2011) use an example of a list of hotels in close proximately to a national park, but the web page did not provide an exact location of this hotel, as the location data (stored on another site) is not shared amongst the web to link the hotel and its location. The semantic web therefore looks to identify the relationship between the national park, the hotel and its exact location by linking these data points.

Sivakumar and Ravichandran (2013) provide a breakdown of the different layers of a semantic web, from the first layer, which consists of URIs and unicode which is used for identification to the seventh layer which consists of 'Trust' and 'Proof', is where the relationships and statements between data can be trusted and proved. This breakdown is given in Table 2.16 and further reading is provided around the other layers that are not discussed here. The layers one, two and three are discussed as they provide the basis for the semantic web design.

Table 2.16: Table of the different semantic layers within the semantic web (Sivakumar and Ravichandran, 2013).

Layer	Name	Description
Layer 1	URI and Unicode	Used for identification of resources
Layer 2	XML	Represent data content and structure
Layer 3	RDF	Describe resources on the web
Layer 4	OWL	Describe the different types of resources and the relationship
Layer 5	RIF	Used for the logical reasoning
Layer 6	SPARQL	Query language and protocol for RDF
Layer 7	Trust and Proof	To verify statements as to draw conclusions and test the trust between the relationships

Following from this, the next section covers the Resource Description Framework (RDF. The RDF is the data model developed to represent the distributed web of data used by the semantic web Allemang and Hendler (2011) and Staab and Struder (2009), also known as the language representation of the semantic web. The semantic web also makes use of XML (eXtensible Markup Language) which allows for the creation of tags, also known as labels to be used on pages or sections of text on a page. It allows for a structure to be created, which can be used by programs to share the 'tagged' data (Berners-Lee et al., 2001).

2.6.3 Resource description framework

The Resource Description Framework (RDF) developed by the World Wide Web Consortium (W3C) is a language in which to develop the semantic web using the XML 'tags'. As stated in Staab and Struder (2009), the RDF was built using previous developments such as the 'Platform for Internet Content Selectivity' (PICS) and the 'Dublin Core' which is made up of triples (statements). It is used to store any semantic data (Sivakumar and Ravichandran, 2013). A RDF triple is in the form shown in Figure 2.34.

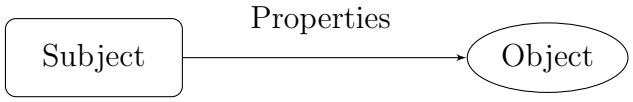


Figure 2.34: A RDF triple consisting of a subject and an object made up of various properties (Staab and Struder, 2009).

The RDF triple is then identified using XML, a type of URI. The *subject* can be an entire web page, part thereof or an entity not directly linked via the specific web page of concern (Candan et al., 2001). The *object* is the value or phrase associated with the *subject* and the *property* is the characteristics of the object associated with the given subject. This could be the type of value, category of the object, *etc.* (Allemang and Hendler, 2011). They provide an example of samples of triples and how different triples can be merged to provide meaningful output given a user request. Different web pages referencing the same *subject* will then use the same URI across the entire web.

The final result of using these triples along with the XML as the URI is the ability to link various points of data on different web pages such that a user can easily locate information relevant to their request, without having to search for the information on different web pages and providing contextual data to a query.

2.7 Benefits and drawbacks of Big Data

Using Big Data Analytics holds great value if used correctly. It can allow for the transformation of economies, and provide productive growth for industry (Philip Chen and Zhang, 2014). This is because analytics assist in improved decision-making, yielding higher desired outputs. In future, using Big Data Analytics will become ubiquitous among enterprises in order to have the competitive edge and will attract employees with the necessary skills in Big Data Analytics (Philip Chen and Zhang, 2014).

According to Philip Chen and Zhang (2014), the challenges that prevent Big Data from becoming ubiquitous within industry are the following:

- Data capture and storage: With the growing number of devices and sensors used by people and industry the amount of data being collected and stored

is continually growing. The CPU-heavy and I/O-poor constraint (CPU speeds increase over time at a faster rate compared to the rate at which data can be read and written) limits the rate at which data can be stored, accessed and processed.

- Data transmission: With the popularity of cloud computing, the network bandwidth speed creates a bottleneck between uploading/downloading data along with security risks due to hacking.

- Data curation: Retrieving, managing, quality assurance and filtering data is a large part of Big Data analytics. With structured data, SQL (Structured Query Language) is widely used as it uses relation data techniques, but with unstructured data, SQL is not able to filter and manage the large datasets. NoSQL (Not only Structured Query Language) is a typical method used to join relational with non-to-near-relation datasets. Employing such data management tools is increasing in complexity as data capturing and storage increases.

- Data analysis: As the amount of data being managed increases, more pressure is being placed on developing more efficient and effective algorithms in order to reduce the processing time of large datasets. *Increment algorithms* are popular as they allow for scalability. Parallel computing is also being adopted to overcome analysis constraints, which is why cloud computing has gained traction as it allows for workloads to be distributed over a large network of parallel computers.

- Data visualisation: Conveying the results in an effective manner with a growing amount of data is another problem attributed with Big Data Analytics. The visualisations need to be scalable and intuitive.

According to Redman (2013), other challenges Big Data faces that prevent wide-spread adoption and integration into enterprises are the following:

- Data quality: Because of data quality varying depending on the source of the data, significant amount of time is spent on 'fixing-up' and filtering the data, in order for an analysis to be carried out.

- Data credibility: When data is sourced and found to be inaccurate or incorrect, this cause the results of the analysis to be invalid thereby defeating the reason for using Big Data analytics. Such instances cause trust in the system and data to evaporate and can lead to less good decision-making.
- Privacy and security: Because of the ever growing global population making use of the internet to provide products and services, large quantities of personal data are being added to the pool of data on the internet. This poses a risk, as the servers on which this personal data is being stored are vulnerable to hacking. With enterprises making use of Big Data Analytics which use/have access to this personal information, security measures need to be put in place to protect the users, bringing with it a large cost component.

2.8 Current features and trends in Big Data

The different trends in Big Data Analytics are investigated, looking at current features within industry as well as future trends for Big Data.

2.8.1 Current features of Big Data systems

Hardware and software combinations in Big Data implementations vary depending on the requirements of the system. Kambatla et al. (2014) outline current hardware and software platforms that are commonly found in Big Data systems. These current features researched by Kambatla et al. (2014) were also similarly outlined and discussed in Butler and Bekker (2017). Current hardware features include:

- Memory and storage: Commonly used for storage, disk drives (having a relatively low cost) with moving parts are used to store data, while using DRAM (Dynamic Random Access Memory) for fast access and caching purposes. The problems associated with traditional hard disks are short useful lifetime and read and write speeds, while DRAM requires static refreshing circuits which consume power during periods of inactivity.

- Processing: With processing, two dimensions of differentiation exist, namely, the performance of each core and the degree of homogeneity of their ensemble. "Wimpy or low power cores which interface with local flash storage balances energy requirements with computational and I/O profiles. Fast Array of Wimpy Nodes (FAWN) is an example of such an implementation, consuming more energy than a dual in-line memory module (DIMM) with two DRAM modules but being 100 times slower than DIMMs. *Amdahl's law* which states that "in parallelisation, if P is the proportion of a system or program that can be made parallel, and $1 - P$ is the proportion that remains serial, then the maximum speed-up that can be achieved using N number of processors is,

$$\frac{1}{(1-P) + (P/N)}\text{"}.$$

  This limits the execution time of processors.

- Network resources: A typical communications stack consists of a link, network and a transport layer with interoperability designed into the stack. Load balancers and application proxies separate internal traffic on a network from external traffic which results in RTTs (Round Trip Times) of network communication to be less than 250 microseconds, where there is no queuing of queries. The performance/throughput of a data centre network is currently hampered by the use of aggregation patterns of large-scale web applications as the response is consolidated back from a many-to-one scenario, causing a bottleneck. Explicit Congestion Notification (ECN) protocols (*e.g.* Ethernet or Infiniband which is an energy-proportional network) are used to counter such a bottleneck by having an end-to-end notification of congestion without dropping packets (stores containing data).

- Energy considerations: The key objective of data centres is to be energy proportional (linear relation between energy consumption and data traffic/load). Batch processing allows for power reductions as loads are known, online/parallel applications with unknown varying loads do not allow for energy-saving schemes such as work consolidation.

When looking at the current software stack solution to large scale datasets, such solutions should i) scale to accommodate large datasets, ii) efficiently leverage hardware platforms and iii) bridge the gap of computing power and data growth. Current software features which best try to meet these requirements include the following (Kambatla et al., 2014):

- Storage systems: Storage platforms are required to be distributed, scalable, elastic and fault-tolerant (be able to process a requests with node failures). Data replication across multiple machines are carried out depending on the availability and fault-tolerance required. The *metadata* (tags) which link the data across the machines require more sophisticated storage models as the amount of data stored increases. NoSQL is common-place for unstructured data distributed over multiple machines.

- CAP theorem: The Partition tolerance, Consistency and Availability (CAP) theorem states that it is impossible for a storage platform to provide guaranteed consistency and high availability with many data partitions. When looking at availability, most applications which do not interface directly with the user can have lower levels of availability. The Google File System (GFS) has lower levels of availability, but is more consistent and has a larger fault-tolerance which allows for scalability. *Bigtable* and *Megastore* are value-stores employed by Google to replicate its data. Open source equivalents are the HDFS (Hadoop Distributed File System) and Hbase which employ Zookeeper, that makes use of the Zab protocol. This is a protocol used by these systems to complete the processing while maintaining consistency.

  Because data systems need to store and recall information, storage systems cannot completely relax the consistency constraint and must therefore maintain a certain level of consistency (ensure when requests are made, the correct data is returned). The less consistent a data storage system however, the more time is required for reconciliation of the data upon request.

- Resource utilisation: Because of the increasing divide between the faster growing data size compared to computing power, resource usage becomes

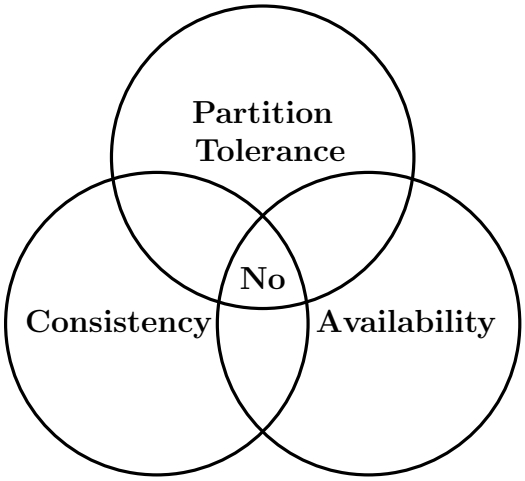


Figure 2.35: The visualisation of the CAP theorem, which illustrates the need to balance the three constraints.

more relevant. This entails having more replicas of data in order to improve availability to ensure the rate of reconciliation (joining the data blocks together) upon request remains adequate. This in turn has an effect on energy usage as more replicas are stored. Using erasure-coding (method of data protection, by breaking data up into fragments to then individually encode and store) to reduce replica file size is a method used to reduce network latency, with the benefit of correcting disk errors in the process whilst offering security. To reduce network usage TCP Nice and Dr. Multicast are techniques available. TCP Nice reduces network usage by using spare bandwidth and asynchronous background communication to avoid interference with regular traffic. *Dr.Multicast* allows IP-multicast to reduce sender/receiver latency.

- Data processing considerations: The Big Data applications are each unique to the input, data access patterns and parallelism of the system. Online stream processing processes each request individually, causing latency. Large datasets as input can batch the I/O to avoid latencies. Applications running in the background run sequentially through datasets while client-end applications would randomly access stored data.

- Data-parallel models: Most data analytics make use of data-parallel models, where the processing of data is done in parallel to increase the amount and rate at which an analysis on the data can be conducted. There are currently three models used to analyse Big Data in parallel, due to the different types of input data arriving at a Big Data Analytics system. These are: *batch processing*, *bulk synchronous parallel* and *stream processing*. The *batch processing* is typically done on static, large batches of data, using the *MapReduce* paradigm (discussed in section 2.4.2) to conduct the analysis. For the same input data as in *batch processing*, the *bulk synchronous processing* (BSP) model is favoured when using *iterative* algorithms to analyse the data. In the BSP model, concurrent computations are conducted in parallel on each node locally. The results are then merged together in a global synchronisation step. Examples of this being used is by Google which has a BSP model *Pregel* which is used for the PageRank system that ranks web pages on Google searches. The final model *stream processing* analyses continually arriving input data. Implementations of this model include Twitter's *Storm* which is able to process user tweets on a large scale, making use of pattern matching to identify trends or events in near real time.

- Data dependent parallelism: Data-parallel programming models being commonly used to conduct multiple computations can be improved through communication/data-sharing across concurrent computations. Communication allows for conflict resolution among computations. Transactional MapReduce (TransMR) allows for transactional executions over key-value stores, typically over Bigtable (a Google database service that provides low latency and high throughput).

2.8.2 Trends in Big Data

With most products and services, improvements, changes or complete redesigns are made to remain competitive or re-invigorate interest. This holds true for Big Data, as further research is conducted in the field, the manner in which Big Data is designed and used changes to ensure the Big Data systems are able to provide

the desired results. The following possible future trends in Big Data are according to Yin and Kaynak (2015) and were also discussed in Butler and Bekker (2017) where the authors outlined these trends from Yin and Kaynak (2015):

- Novel technique improvements in data mining: Using different machine learning tools to solve unique problems.
- Cloud-based storage and transmission: Using cloud-based systems to store and transmit the data from-and-to the processing nodes. Eliminating possibilities of data loss.
- Solution focused on data monitoring and control: Development of systems to monitor data quality, *etc.* when working with Big Data. Thereby ensuring the data being collected and stored is of value, reducing overheads required for data cleaning, *etc.*
- Plant-wide optimisation and prognosis: Utilising Big Data to manage and improve the processes within large scale production or distribution plants.
- Supply chain and risk management systems: Utilising Big Data to mitigate or reduce potential issues experienced throughout a supply chain, by identifying potential problems given a set of factors (identified previously).

2.9 Synthesis of literature

For this project, the goal is to provide the industrial engineering community with a demonstrator of Big Data and Big Data Analytics. This Demonstrator had to make use of open-source Big Data software and commodity hardware, from which to illustrate the benefits of Big Data Analytics over a traditional analytics system (not using Big Data software), and provide a starting point for further research in the field. Therefore the literature review was done in order to gain an understanding of Big Data, and then what was required in order to develop the Big Data Analytics Demonstrator. The literature review started by providing an understanding of what is currently available and published around Big Data, from the definitions to technologies included under Big Data. This included how Big

Data is defined, using the four Vs, which describe how different characteristics together form Big Data. The significant components that were identified were *Volume* and *Velocity* from literature. Due to current technological development and usage, more data is being generated at a faster rate (from tweets on Twitter to ideas being shared on platforms such as Reddit), which needs to be analysed, utilising these aspects of Big Data. Using these qualities, this would ensure that the Demonstrator was able to store and analyse large volumes of data quickly were deemed important features (qualities) of a successful Big Data Demonstrator.

The next phase of the review (after establishing how Big Data was defined), was to research the architectural *structures* that would be required in order to develop a Demonstrator of Big Data, which would be able to analyse large volumes of data in a rapid manner. As was discovered, there were many architectures used in industry available that were developed for a specific application and which use Big Data. Of these, a selection of architectures were further investigated. From the investigation, commonly shared attributes across all were discovered namely, a *data storage*, *data analysis* and *results interpretation* phase.

To compare industry architectures with those from literature, reference architectures were then researched. Reference architectures provide guidance on the different components that together make a successful architecture and are therefore used to develop industry architectures. The architecture from Maier (2013), a reference architecture purposefully developed for Big Data related projects, was therefore also further studied. When comparing the reference architecture with what was found in industry, the similarities were identified. These included having a *data storage* system, where data that had been gathered was to be integrated and stored in a single data warehouse which distributes data within and from which it could be accessed. The architectures also had a manner in which the data warehouse would manage and protect the data in the warehouse. The components responsible for analysing the data, were also similar, as all included systems in place to analyse data using data mining and machine learning. The results from the data mining and machine learning were then output to the user or analyst to make relevant decisions through a form of visualisations and reporting. Considering these shared attributes between Big Data reference and industry ar-

chitectures, it was determined that these three components were relevant features to include in this project's Demonstrator architecture.

After identifying and establishing the qualities (processing *Volumes* of data at a high *Velocity*) to include in the Demonstrator, along with the relevant architectural components of a Big Data Analytics system, the next phase of the literature review was focused on the systems that would need to be included to store, analyse and interpret large volumes of data rapidly, to fulfil these architectural requirements. Because the scope and goals of this project included that the Demonstrator make use of open-source (free-to-use) software, the research did not include any pay-to-use Big Data software solutions. The Hadoop project's software was determined to be an appropriate solution to the storage of data. For the storage component of this Demonstrator, specifically the HDFS which, provides the means to store data across multiple nodes in a fault-tolerant manner was chosen. Other Hadoop projects were available, but due to the project scope the storage solution did not require a system that would, for example, assist in near real-time analysis.

For the analytics system of the Demonstrator, research was conducted on various available solutions which would be able to analyse large volumes of data in a rapid manner. The first was considering the MapReduce framework, as it is part of the Hadoop project. Thereafter another solution was researched, which is growing in usage, namely Spark. The Spark software is open-source, regarded as one hundred times faster than MapReduce, providing the necessary data mining and machine learning tools to conduct an analysis, and has support for HDFS storage.

The goal of this project's Big Data Analytics Demonstrator (BDAD) is to illustrate the benefits of large scale analytics, meaning that an analysis that is conducted is a major component of the system, as without the use of appropriate analytic tools, the necessary results could not be generated. The next phase of the literature review therefore required a better understanding of the different analytic tools that are traditionally used in Big Data Analytics. The research into Big Data Analytics showed there was a gap in the definitions and understandings of what encompasses Big Data Analytics. Because of the differing viewpoints of what is included under Big Data Analytics, the research group USMA, which the

researcher is a part of, held a focus group meeting to consolidate the different understandings of BDA, and from this developed Figure 2.10. Under this definition, Big Data Analytics was deemed to consist of three processes, each process consisting of steps which together take data and ultimately provide newly gained insights, under which data mining was a single step or phase, and machine learning was applied in data mining to mine the data for relevant trends, patterns *etc.* to gain these insights. The research conducted on the analytic processes showed that the KDD process was the original process on which the others were built, and it is generic such that it could be modified as desired. For this reason, it was chosen to be used during the Big Data Anaytics process, after the BDAD had been developed. The final part was to understand how these machine learning tools were able to provide the large scale analytics and therefore under each tool, a selection of commonly known techniques was further researched to understand their working.

From this, the qualities of the BDAD, the components required for the architecture from which the Demonstrator would be built, the different software solutions to analyse and store the data, and finally the processes, tools and techniques were identified, which would allow for the successful analysis of data by the Demonstrator. The gap in the understanding of Big Data Analytics (BDA) was identified during the literature review, whereby the different understandings were consolidated and outlined. Other research not directly applicable to the Demonstrator, but part of the Big Data environment, was on how data is shared and contextualised to ultimately provide the online Big Data environment. A final consideration was to research the trends in Big Data, ensuring the Demonstrator would be in line with current features and trends, providing a spring-board from which research into these trends could be pursued in future or expanding upon the current trends.

Next, a brief summary is given of each of the sections researched within the literature review.

2.10 Summary of literature

The literature review firstly included the definition of 'Big Data'. Big Data is defined by multiple characteristics such as *volume*, *veracity*, *velocity* and *variety*. Research was then conducted on Big Data Architectures in literature and industry, how these architectures were developed, and what is included in the architecture. The focus was specifically on what is required in a Big Data Architecture and Big Data reference architecture. The reference architecture serves as a guide to what components are typically included in any architecture that uses Big Data. Thereafter, the focus shifted to Big Data Analytics, the methods KDD, SEMMA and CRISP and machine learning tools and techniques used to analyse large datasets. This was after holding a focus group with members of the USMA research group within Stellenbosch University's industrial engineering department, on how Big Data Analytics was defined and what is included under this term. After establishing the understanding of BDA, detailed research into the machine learning tools and commonly used techniques are discussed. Within each technique its applicability to different applications was outlined. An overview was given of Hadoop, the HDFS storage system and analytics method MapReduce, after which the Apache Spark analytics software and its workings were discussed as well as benefits over MapReduce. The semantic web and RDF was then researched, its applicability towards the Big Data environment and how it has helped shape the expansion of Big Data. Lastly, the benefits and drawbacks, current features found in Big Data, and trends were researched to identify possible areas for further research. These include cloud-based storage and transmission, incorporating Big Data into optimisation of production plants, or using Big Data in supply chain systems to identify areas of concern.

Using the information from this chapter, a proposed Big Data Architecture will be developed by the researcher which is to be used during the development of the Big Data Analysis Demonstrator in the next chapter. Thereafter, during the development stage, the applicable processes, tools and techniques required to develop the demonstrator will be applied and discussed further in Chapter 4. The literature review performed in this chapter therefore fulfils Objective I, as previous implementations of Big Data, methods used in Big Data Analytics,

technologies used in Big Data Analytic systems, as well as the trends in Big Data were researched.

Chapter 3

Proposed Big Data Architecture

A literature study was conducted in Chapter 2. The literature review was undertaken to understand what is Big Data and how it was defined. Along with this, to research Big Data Analytics (BDA) and what falls under this definition (machine learning *etc.*). The final large components were to research what architectures for Big Data have been developed and finally what systems and methods there are to conduct Big Data processing (Hadoop, MapReduce and Spark).

The chapter is outlined as follows, firstly, the methodology that was followed to develop the architecture, by Rozanski and Woods (2005) is discussed. The methodology to visually communicate the architecture components is then expanded on. The goal of the architecture is clearly outlined, and the stakeholders involved in the project are identified. The different components to be used by the demonstrator are expanded upon, from which the architecture is to be developed. Finally, the developed architecture is evaluated and validated by comparing it to the architectures in the case studies in section 2.2.4, which were recently developed, found in literature and used within industry to ensure the required structures are in place to develop Big Data Analytic systems.

3.1 Methodology used to develop the Big Data Architecture

The methodology used by Rozanski and Woods (2005) is used in the development of the proposed architecture along with overlapping steps outlined by Maier (2013), which will be highlighted and included here. The methodology is appropriate as it is applicable to software systems architecture. The architectural elements as stated by Rozanski and Woods (2005) are:

- Clearly defined set of *responsibilities*
- Clearly defined *boundary*
- Clearly defined *interface*.

The *responsibilities* attribute relates to the stakeholders involved in the architecture and the project as a whole. These stakeholders need to be identified and there should be clearly defined how their needs are to be met, as in Maier (2013) as well. The scope and architecture goals shape the *boundary* of the architecture, which are formed by examining the requirements of the stakeholders. The *interface* attribute is concerned with how the stakeholders are to interact with the system.

The methodology from Rozanski and Woods (2005) is outlined as follows:

1. Defining the scope and goals
2. Identifying the stakeholders
3. Identifying the architectural scenario
4. Selection of the architectural style
5. Developing the architectural model
6. Documenting the architecture
7. Validating the architecture.

After identifying the stakeholders and their needs, thereby establishing the architecture goals and scope as described by Maier (2013), the architectural scenario has to be outlined.

The architectural scenario can be used to clearly outline the use cases to be experienced, thereby providing a means to clearly define the architecture and its goals, similar to Maier (2013). The scenarios under which the system needs to run also provide a means of evaluating and validating the architecture (Rozanski and Woods, 2005). The architectural style is concerned with the schema by which the system and all its components are communicated. The next step involves developing the architectural model, ensuring the goals are included (Maier, 2013).

The documentation of the architecture is concerned with ensuring the stakeholders are provided with relevant documentation demonstrating the success of the architecture. The different properties that Rozanski and Woods (2005) have to be included to provide effective architectural descriptions are given by Figure 3.1 (Rozanski and Woods, 2005).

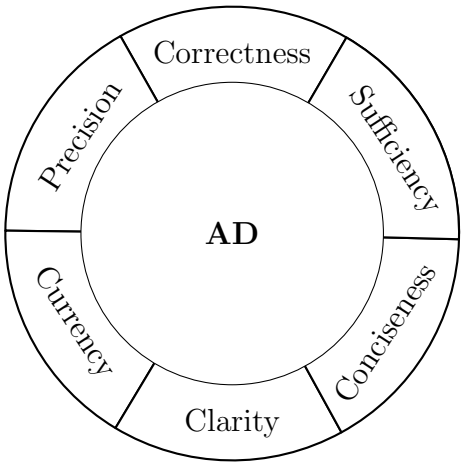


Figure 3.1: The different properties that are included to provide effective documentation of the architecture description (AD) (Rozanski and Woods, 2005).

The *sufficiency* means ensuring there is enough detail in the architecture to communicate the different views and perspectives. *Correctness* is whether the architecture represents the needs of the stakeholders, *conciseness* is ensuring the key features are included, but it is noted that this is however subjective. *Clarity*

is ensuring the model and all the components are able to be easily understood. *Currency* ensures that the architecture evolves due to changing requirements, which need to be reflected in the model. The *precision* is ensuring the model and all its components describe the system with sufficient detail.

The final step is evaluating and validating the architectural model, according to Maier (2013). As stated by Rozanski and Woods (2005), this step allows for abstractions made of reality to be tested, as well as the checking of errors, selling the architecture to stakeholders to assess how it meets their needs and also validating the assumptions made.

Next, the methodology's steps outlined are applied in the development of the Big Data Demonstrator Architecture.

3.2 Big Data Demonstrator Architecture

To provide a basis from which to develop the Big Data Demonstrator, an architecture is required to guide the design process as described in section 2.2.1. The architecture needs to meet the goal set out by the project requirements and stakeholders involved in developing and using the demonstrator identified in section 3.2.1.

This section is divided into four major subsections which follow the methodology outlined in section 3.1. The architecture scope, goals and stakeholders are defined, the architectural style is briefly discussed, the model development and the components (all together forming the architectural system) thereof are then discussed, making use of what was learnt from Marz and Warren (2015), Galster and Avgeriou (2011) and Maier (2013). The final section concerns the evaluation of the proposed architecture, by comparing it to case studies of recent industry implementations and literature to validate.

3.2.1 Stakeholders of the Demonstrator Architecture

As stated by Taylor et al. (2010) and Rozanski and Woods (2005), the stakeholders of a software architecture need to be identified. The different stakeholders

for this proposed architecture were derived from the comprehensive list of typical stakeholders in a software systems architecture, provided in Table 3.1. The stakeholders' needs are what drive the development of the architecture and its goals, as these needs had to be met successfully to produce the final system.

Table 3.1: A table including the different stakeholders that are involved in this project and its architecture, which is a modification of the table from Rozanski and Woods (2005).

Stakeholders	**Description**
Acquirers	Not applicable to this research project as the demonstrator is not funded and developed for a public or private organisation
Assessors	Supervisor, Industry partner and the project examiners.
Communicators	Researcher of the project
Developers	Researcher of the project
Maintainers	Not applicable as the system is a research project and not to be deployed in industry. However, possible future researchers of related Big Data projects can be stakeholders
Suppliers	The researcher of the project who is supplying the hardware and software on which the system is to be developed and run
Support Staff	The researcher and possible future researchers can provide the supportive role once the demonstrator is developed
System Administrators	The researchers, supervisor and possible future researchers will run the system/demonstrator once developed
Testers	The researcher, supervisor and industry partner are to test the demonstrator to determine it has met the requirements
Users	The researcher, supervisor, industry partner and possible future researchers are all users of the demonstrator

The *acquirers* are the persons responsible for the procurement process of the system or product. There are no acquirers as this project is a research project. The *assessors* ensure the project meets the requirements and standards set out by an organisation or law, and test the system for this. The *communicators* need to convey the system and its rationale, including all technical and non-technical aspects to other stakeholders. The *developer* is concerned with the construction and deployment of the system and as stated from Rozanski and

Woods (2005), who take the software through the software development life cycle (design, code, test and accept). The *maintainers* ensure the system's operation once deployed, such as debugging, preservation of the knowledge and also manage the change control (evolution of the system over time). The *suppliers* provide the required hardware and software on which the system is to operate and can impose constraints due to the hardware or software that need to be factored into the system. The *supportive staff* provide support to resolve issues users have with the system. *System administrators* are the stakeholders who deploy the system once it is developed. Their focus is on disaster recovery, business continuity, monitoring, resilience and scalability of the system. The *testers* run tests on the system to ensure it meets the requirements set out, thereby establishing if it is suitable for deployment. The testers are independent from the developers to ensure a thoroughly objective evaluation is conducted. The *users* as stated by Rozanski and Woods (2005), define the system's functionality and use the system itself. Functional concerns of the users have to be met, as well as operational (security and performance) and are what help shape the goals of the system.

3.2.2 Big Data Demonstrator Architecture goal

Using the *problem statement* and *project scope* discussed in Chapter 1 and stakeholders, the demonstrator and therefore the architecture on which it is built; need to be able to use current technologies available in the Big Data environment. This is in order to conduct a batch analysis of a large dataset (structured, semi-structured or unstructured data) on commodity hardware with low latency. The BDAD should though this, provide a demonstration to the industrial engineering community, the utilitarian nature of Big Data, in providing BI which is superior to that of standard analytics (when working with Big Data).

3.2.3 Big Data Architecture development

The architecture proposed is not an original conceptualisation, but makes use of different reference architectures (RA), by Maier (2013) and by using components of the *Lambda Architecture* developed by Marz and Warren (2015) in section 2.2.3. The RA by Maier (2013) is used, because it was developed specifically for Big

Data related projects, and was deemed applicable to this project, as it includes a wide variety of components. The *Lambda Architecture* was chosen due to its use in industry and literature, an example of this is the architecture developed by Astakhov and Chayel (2015) and Xhafa et al. (2015). The flexibility of the *Lambda* architecture in that the *stream processing* layer need not be included as set out in the scope, is favourable as the demonstrator is to conduct only a batch analysis. Together with this, and the design of the *Lambda Architecture* which ensures systems with low latency, high accuracy and fault tolerance were favourable qualities applicable for this project, and thus the proposed architecture.

Some of these components are integrated into the architecture and system, and cannot be visually depicted, such as *privacy* or *Manage data quality/uncertainty*. The *Metadata management* and *Data lifecycle management* for example, are to be handled by the HDFS *NameNodes* discussed in section 2.4.1. The *Data distribution* is handled by the HDFS format itself to duplicate the data across the multiple nodes and is also not shown in the architecture itself. The *stream processing* component as stated previously falls outside the scope of this project and is not included.

Thus, it can be concluded that the proposed architecture is a hybrid approach, making use of different components and methods found in literature.

The architectural style as stated by Rozanski and Woods (2005) needs to be chosen to ensure the architecture is easily communicated to the stakeholders involved. To do this, the *Object-Process Methodology* in Dov is to be used.

This method is also applied so as to ensure the *architectural descriptions* given are fulfilled. The reason given in Butler and Bekker (2017) as to why the OPM is used to communicate the architecture components proposed there, applies to the development of the architecture proposed in section 3.2.4. As stated in the article by Butler and Bekker (2017), it is a unified and well defined method for systems development, a framework that is developed in order to effectively communicate a systems architecture. The OPM makes use of graphical and natural language means to communicate the architecture and is an ISO standard (ISO/PAS 19450:2015). The OPM makes use of objects which are static and undergo a change due to particular processes being applied, thereby changing into a different object (state). The objects typically store data, queries *etc.* and

the processes are executable steps or pieces of code which ingest data from objects to deliver results or new data, thus a new state is formed.

Next, using the goals, stakeholder requirements and the requirements for the architecture in section 2.2, the proposed architecture and components (objects, processes and states) are discussed at a high level. The components work together in order to provide analytical results to the analysts' queries.

3.2.4 Proposed Architecture model and components

The proposed Big Data Analytics Demonstrators architecture is given in Figure 3.2. The demonstrator was developed, taking into consideration the requirements set out by Maier (2013), which include

- Data Extraction
- Stream Processing
- Information Extraction
- Manage data quality/uncertainty
- Data Integration
- Data Analysis
- Data Distribution
- Data Storage
- Metadata Management
- Data Lifecycle Management
- Privacy.

The architecture as shown illustrates how data is taken from the *data source*, and run through a *HDFS Converting* process to store the data in a *HDFS* repository, similar to the architecture developed by Butler and Bekker (2017) and upon which this architecture is based and expands. Then given the queries submitted by an analyst, the *Data Analysing* process is executed whereby the *Query Results View* stores the results and the *Visualising* process is completed to provide the results of the query visually to the analyst. This finally provides a means to gain BI through the analysis conducted.

The following is a further discussion of each of the Big Data Analytics Demonstrator architectural components.

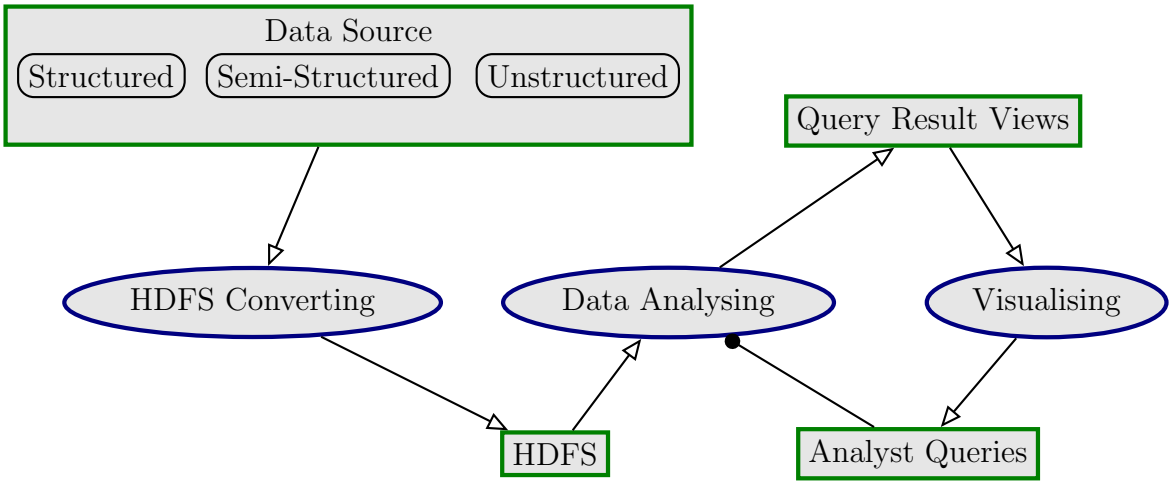


Figure 3.2: The proposed Big Data architecture for the demonstrator, high level System Diagram 1 (SD1) view.

3.2.4.1 Data source

The *Data Source* object is the point at which the data is received from various sources. For this project, the data is to arrive at the demonstrator in a raw format, but all sensitive data removed (anonymised) if needed. The *Data Source* allows for any data type to be accepted into the demonstrator: structured, semi-structured or unstructured data. This allows for flexibility in the type of analysis to be conducted in future, but for this project, as previously stated, only *structured* data is analysed in a *batch* analysis manner.

3.2.4.2 HDFS converting

The process of converting the input data from the *Data Source* object is conducted as the data is collected in batches. The goal of this process is to take the input data and then convert the data into the HDFS format. A 'data warehouse' is then created which stores this data in the HDFS format. A more detailed view of the process undergone within *HDFS Converting* is given in Figure 3.3.

Data is sent to the *HDFS Converting* process from the *Data Source*, whereby first a *Data Selecting* process is undergone. This process is to select appropriate datasets (if applicable). Otherwise, the datasets are converted to the HDFS

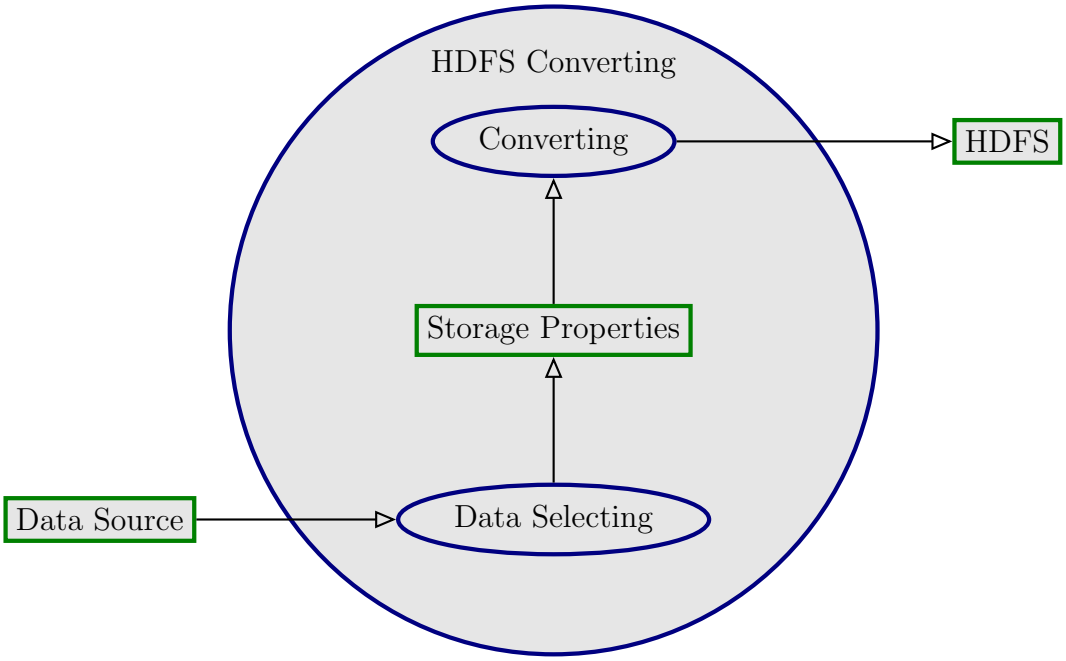


Figure 3.3: A zoomed-in view of the *HDFS Converting* process where the data is taken from the data source and converted into the HDFS format and stored, showing the System Diagram 2 (SD2) level view.

format by firstly specifying *Storage Properties* (number of replications and data-block size) and then converting the data into the HDFS format given by a *Structured Data Store* object HDFS. To perform this conversion, the *Hadoop Flume* or *Hadoop Sqoop* software are considered for relational databases. *Hadoop Flume* and *Hadoop Sqoop* are both open-source software solutions to convert relational database datasets into HDFS format. If however, the data is not from relational databases, the data is then converted into the HDFS format without using a software solution, making use of the built-in Hadoop commands.

3.2.4.3 Data analysing

The *Data Analysing* process is where the data stored on the *HDFS* store is processed and results generated.

The process shown in Figure 3.4 is initiated by the analyst submitting queries to be run on the data in *Analyst Queries*. The *Information Extracting* process

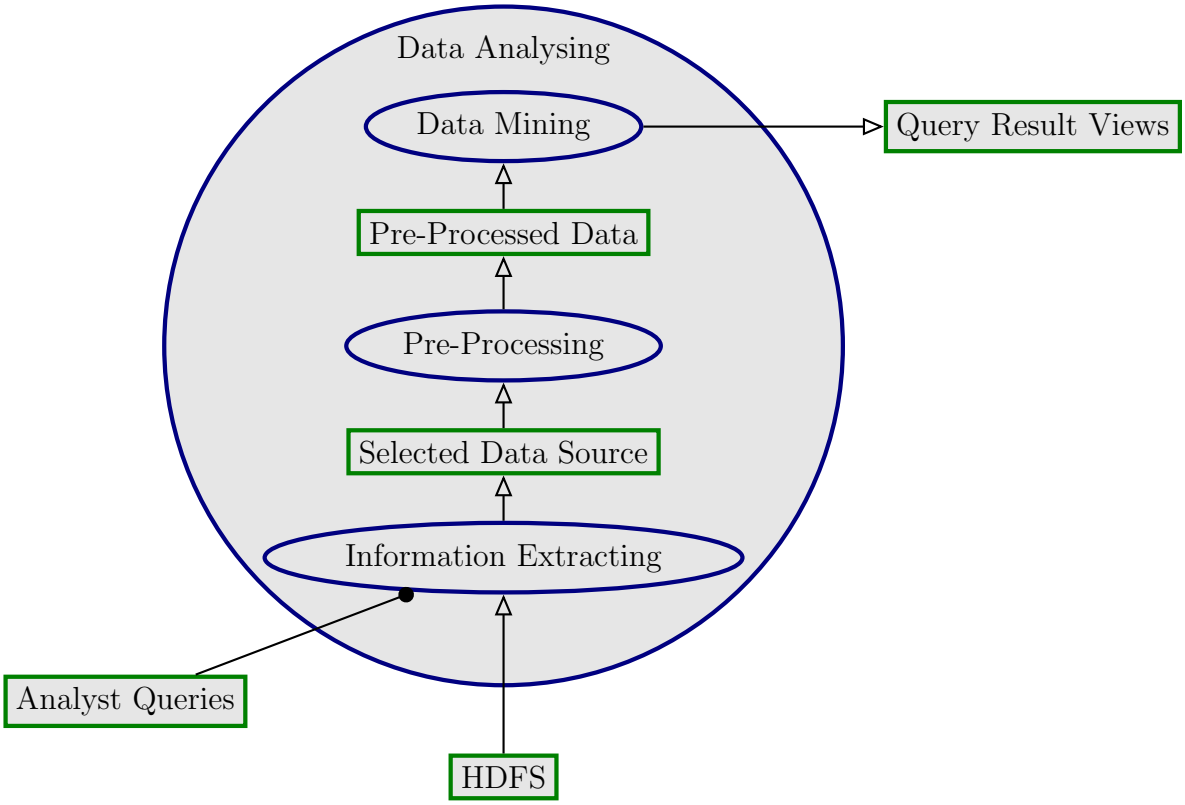


Figure 3.4: A zoomed-in view of the *Data Analysing* component where data and queries are consumed, after conducting *preprocessing* and *data mining*, the results are then stored in *Query Result Views*, showed in the System Diagram 2 (SD2) level view.

then selects the relevant dataset to be queried, represented by the *Selected Data Source* object. This data is then run through a *Pre-Processing* process. The *Pre-Processing* involves filtering, sorting, noise removal and imposing of a given format readable to machine learning algorithms, to be used in the following *Data Mining* process. During this process, data mining activities as discussed in section 2.3, are executed. The analytics software employed in this component of the architecture is the Spark software, which was discussed in section 2.5. The Spark software provides the tools in order to conduct the pre-processing and processing of the data in a rapid manner, as required for this project.

The results from the *data mining* process are then stored on the *Query Result Views* object. If the analyst requires the results to be visually illustrated the

visualising process is run, discussed next.

3.2.4.4 Visualising the results

This process is to provide the analyst with a means to illustrate visually the results from the analysis that has been conducted. The resultant visualisations of the data, graphs, charts, concentrations *etc.* are used by the analyst or researcher to develop a report on the results obtained. The report can then be shared amongst other stakeholders, providing the BI for effective decision-making.

The visualisation software used for this project is a combination of free-to-use software namely, *matplotlib* and *plotly*. These visualisation packages allow for the data to be displayed in tables and various plots. The software is chosen due to its open-source nature and its ability to quickly provide the visualisations of data required for this project. A further discussion of this software is given in section 4.2.5.

3.2.4.5 HDFS storage

The *HDFS* object shown in Figure 3.2 is the 'data warehouse' that is used by the demonstrator to store the batch dataset after conducting the *Data Integrating* process. The data, of any data type, is stored in the HDFS format (structured, semi-structured and unstructured data types) allowing for the demonstrator to be flexible when analysing different data types from different sources. The storage of the data is to be distributed across multiple nodes to allow for a distributed analysis, redundancy and fault tolerance of the system.

3.2.4.6 Query result views

After conducting the analysis according to the query submitted by the analyst, the results are stored separately. From this storage, the analyst can view and edit the results directly (if necessary). This storage is however a temporary storage, as new analyses are conducted, the old results are overwritten. This is to lower the storage costs of the system, and if required the analyst can export the results to prevent it from being overwritten. After this, the analyst can then run the

results through the visualisation process in order to visually display and analyse the data, typically through graphical means.

3.2.4.7 Analyst queries

The queries submitted by the analyst are handled by the *Analyst Queries* object. This is similar to the *Query Result Views* object which temporarily stores the analyst's queries and each query is then executed by the *Data Analysing* process on a first-come-first serve basis. After the results have been generated and the analyst requests the data to be visually illustrated, thereby completing the *visualising* process, the visualised results are returned to the analyst and the specific query.

3.3 Evaluation of the Proposed Architecture

The following evaluation of the proposed architecture was completed by analysing and comparing the architectures and architectural components researched in sections 2.2.2, 2.2.3 and 2.2.4. The proposed architecture as shown in Figure 3.2 is comprised of three main processes, each of which contains sub-processes.

Before continuing with the evaluation, an overview is provided firstly of the Lambda Architecture, as it is a large component of the proposed architecture. As discussed in section 2.2.3, the architecture consists of three layers, a *batch*, *speed* and *serving layer*. The speed layer was not considered for this project, due to the project scope and goals. The batch layer uses either incremental or recomputation algorithms to generate the views (results) from an analysis. The results or views are then sent to the serving layer, which can be queried by the analyst. These two components were employed by the proposed architecture. The evaluation of the proposed architecture now follows.

Table 3.2 provides an outline of the different components of the proposed architecture, and which of these were also used in the Lambda Architecture (LA), reference architecture (RA) and case studies, as a brief summary in order to compare and thereby validate the proposed architecture.

Table 3.2: Table of the different components used in the proposed architecture, and from which sources in literature and case studies these components were derived. Using the Lambda Architecture, reference architecture and case studies.

Proposed Architecture		**Architectures from Literature**
Component	**Function**	**Component and Source**
Data Source (object)	The Data Source, data of different types are collected	Data Source in RA Maier (2013), New Data Stream in LA Marz and Warren (2015), persistent storage in Xhafa et al. (2015), HDFS storage in Mayilvaganan and Sabitha (2013) and storage module in Li et al. (2017)
HDFS Converting (process) and HDFS	Selecting the different data sources and storing the data in HDFS	Data Extraction, Data Storage, Data Integration, Data Distribution, Metadata Management in RA Maier (2013), New Merged Data Store in LA Marz and Warren (2015), Distributed Data Collection System in Marchal et al. (2014), Persistent Storage in Xhafa et al. (2015), HDFS Storage in Mayilvaganan and Sabitha (2013) and Storage module in Li et al. (2017)
Data Analysing and Query Result Views	The relevant data source is extracted, pre-processed and analysed. After, the results are then stored	Pre-compute Views in Marz and Warren (2015), Information extraction and Data Analysing in RA Maier (2013), Model Learning in Xhafa et al. (2015), Batch Layer in Astakhov and Chayel (2015), Analysis Module in Li et al. (2017)
Visualising	The results from the analysis are visualised, to conduct interpretation thereof	Batch Views in Servicing Layer of LA Marz and Warren (2015), Model view Xhafa et al. (2015), Visualising under the Iaas of Li et al. (2017)
Analyst Queries	The specific query that is submitted by the analyst to perform an analysis	Used by all, but not shown, as all require a query to be analysed, Maier (2013), Xhafa et al. (2015), Mayilvaganan and Sabitha (2013), Marz and Warren (2015), Li et al. (2017), Astakhov and Chayel (2015), Marchal et al. (2014)

Comparing the work in section 2.2.2 from Maier (2013) for the first process, *HDFS Converting*, the proposed architecture does include various of the components outlined therein, which form part of a Big Data Analytics (BDA) system, shown in Table 3.2. The HDFS storage system used in the *HDFS Converting* process provides these components by integrating the data into a single file system, which is distributed and stored across multiple nodes, while conducting the metadata management automatically. This is also in line with the Lambda Architecture described in section 2.2.3, where data are gathered from various sources and then integrated into a single batch storage system and location. Taking into account the different case studies included in section 2.2.4, most include a system where the first step includes a central system which distributes and stores data gathered and integrated from different sources. Therefore, as shown, the first process (component) of the architecture does include a relevant method found in literature and industry, to integrate and distribute the data storage when new data has been gathered.

Evaluating the second component or process namely *Data Analysing* of this projects proposed architecture, there are three subprocesses contained, as shown in Figure 3.4. The *Data Analysing* process uses the Spark analytics software in order to conduct the various subprocesses contained within the Data Analysing process. This software combines the selection of the relevant dataset, then conducts the necessary pre-processing to provide the ML algorithms with the queried data in the required format. From this, the software is then able to conduct data mining, which uses various ML algorithms available in the Spark suite. Comparing this to Maier (2013), the project's architecture includes a *Data Analysing* component found in the reference architecture of Maier (2013). In the *Data Analysing* component, there are deep analytics components included, which use ML to mine data, this is in line with this project's architecture. When comparing this project's *Data Analysing* process with that of the Lambda Architecture, this project performs an analysis of the data in batch recomputations as new data is analysed with the entire dataset. Comparing the *Data Analysing* component of this project to the different case studies, shown Table 3.2, it could therefore be a valid component to be included in this BDAD. An example of a case study is Li et al. (2017), where different modules are outlined which together form the

Platform-as-a-service. For this architecture, Li et al. (2017) do differ however, in that data is first cleaned and pre-processed before stored, and from storage, the data is then analysed using ML. Therefore it can be concluded that the *Data Analysing* component of this project includes the required components for a Big Data (BD) architecture, as found in literature and industry.

The final process included in the architecture is the *Visualising* process. This process was included to provide the necessary visualisations (graphs, tables *etc.*) in order to communicate the results of the analysis to the analyst. Considering the reference architecture in section 2.2.2, this is equivalent to the Data Distribution component in which an analyst is provided with reports and results which are separate from the analytics system. This is in line with the *Visualising* process, where results are consolidated and graphically output to the analyst on a single system using different software from that used by the analytics system. Evaluating the *Visualising* component against the Lambda Architecture, the results are equivalent to the views that are generated; however, this project does not store these views or results as in the Lambda Architecture, but requires rather recomputation to visually output the results. Considering case studies from industry, most of the architectures outlined included a visualisation component as found in this BDAD proposed architecture, where the results from an analysis are sent to a separate process where visualisation and interpretation of the results is completed. This is therefore in line with the proposed architecture.

Considering the reference architecture, the Lambda Architecture and the various case studies, it can be concluded this project's proposed architecture, with each of the individual components, provides an architecture that is in line with a typical BD Architecture found in literature and industry as most components that constitute a Big Data (BD) Architecture are included. The proposed architecture developed is therefore a suitable framework to develop the Big Data Analytics Demonstrator (BDAD), which could use open-source Big Data technologies to develop a Big Data Analytics system, fulfilling Objective II as set out in Chapter 1.

The next chapter considers the demonstrator that was developed using the proposed architecture, and open-source software outlined in Chapter 2 namely, Hadoop's HDFS and Spark to store and analyse Big Data, after securely coupling three computers together over the Stellenbosch University network.

Chapter 4

System Design and Development

For any analytics system, an architecture needs to be designed which can be used to outline and describe the working of each component functioning together in the architecture, to achieve a specific goal. In the previous chapter, the methodology by which to develop such an architecture was researched and outlined.

Using the *Lambda Architecture* and modifying it to the project requirements and scope, an architecture was developed to be used in the development of the Big Data Analytics Demonstrator. The different stakeholders that are involved in the project and its architecture were also identified and discussed along with the scope and goals. Using these requirements as a guide, the architecture and its components were developed using the *Object Process Methodology* (OPM). The OPM is a well defined method for systems design, and is also an ISO standard (ISO/PAS 19450:2015). The methodology uses graphical and natural language means to communicate unambiguously with a systems architecture.

In this chapter, the different components of the Big Data Analytics Demonstrator are presented as set out in the architecture. The first section outlines the hardware that was available to develop the demonstrator and why it was chosen. then, how the computers were connected together in order to develop the Big Data Analytics system, and then secure the connection, is discussed.

In the final section, the reasoning is provided for using a specific operating system on which the Demonstrator runs, as well as why a specific programming language, data storage, data analytics, and visualisation solutions were chosen.

The scope of the project means that the system must be able to analyse *structured* data, make use of commodity hardware, be scalable, fault-tolerant, make use of machine learning to analyse the data and demonstrate the core aspects of Big Data, defined by the four Vs.

4.1 System hardware

The computers used in the development of the Big Data Analytics Demonstrator are outlined in this section. This includes different components and how they were connected together securely.

It should be noted that at the time writing of this thesis, the computers that were used are not proposed or necessarily recommended to be used, as they have a set performance capability. The performance of the computers will have declined over time, due to the progress made in computer technology. The computers were used only because of their availability, as well as being commodity hardware which is in line with the scope and goal of this project. Any other commodity hardware available at the time of development, can be used to develop a Big Data Analytics system.

4.1.1 Hardware selection

For the Big Data Analytics Demonstrator developed in this project, three computers were used. The first computer was to be used as the *NameNode* (master node) and two further computers were the *DataNodes* (slave nodes). The master node is where the programs were developed, and from which the analysis is deployed. From this node, the analyses were managed, and the results displayed and interpreted. The slave nodes are configured to conduct the pre-processing, computations and analysis of the data. Figure 4.1 shows how the three computers are connected together on the Stellenbosch University (SU) network, where this project was developed.

All computers used for this project were Dell computers, the computer specifications are, as from Dell (2018a) and Dell (2018b):

- The master node specifications:

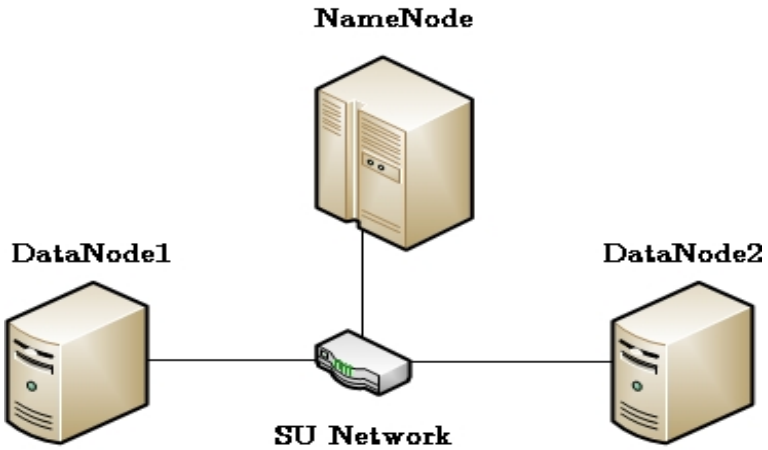


Figure 4.1: A visual representation of the hardware used in the projects Big Data Analytics Demonstrator, and how the computers were connected together.

1. CPU: Intel 4th generation Core i7-4770 with a clock speed of 3.4GHz.
2. GPU: Integrated Intel HD Graphics 4600.
3. RAM: Non-ECC dual-channel 1600MHz DDR3 SDRAM, with 24 GB of memory.
4. Networking: Integrated Intel I217LM Ethernet LAN 10/100/1000; supports optional PCIe 10/100/1000 network card.
5. Storage: One 500 GB disk drive (ST500DM0 02-1BD142 SCSI Disk Device) and one *Solid State Drive* (SSD) of 256 GB (SAMSUNG SSD SM841N 2.5 SCSI Disk Device).

- The specifications of the two slave nodes:

1. CPU: Intel Xeon 5500 series processors, Intel(R) Xeon(R) W3550 with a clock speed of 3.07GHz.
2. GPU: Support for 2 PCI Express x16 Gen 2 graphics cards up to 150 W, NVIDIA Quadro 4000.
3. RAM: ECC 1333MHz DDR3 memory configured to 12 GB.
4. Networking: Integrated Broadcom 5761 Gigabit Ethernet controller with Remote Wake UP and PXE support.

5. Storage: One disk drive with 500 GB of storage (ST3500418AS ATA Device)

Next, how the computers were connected to the university network is discussed.

4.1.2 Coupling the computers to a network

The following steps and considerations were taken during the BDAD system development, where three nodes were linked together. The nodes could either have been connected together using a local network or an existing network that is already in place (in the working environment). Another alternative was to use a WiFi or Ethernet connection to connect the nodes together over a network. Considerations needed to be given to the type of connections. With WiFi it has to be checked to ensure that network traffic is low (so that the nodes do not have to compete with other computers for bandwidth on the WiFi connection), while with both Ethernet and Wifi, it should be ensured that the network has enough bandwidth. Bandwidth refers to the volume of data able to be transferred within a given time; the higher the bandwidth, the more data can be sent and received at a rapid rate (Fisher, 2018). In this project, an Ethernet connection was used as it offered low traffic and high bandwidth of 100 megbits/second (available at the time of this project), which is desired when working with Big Data. To check if the nodes are connected to the network, in the network settings, the output is provided, as shown in Figure 4.2, which indicates that the specific master node is connected to the network.

For this project, the existing university network was used, which is indicated in Figure 4.2 by the specific IP address shown. The following is an overview of the steps undergone to establish, and then create a secure connection between these three nodes, which were connected together on the University network. A detailed discussion is given in Appendix A in section A.1:

1. The first step involved identifying the individual nodes on the University network. This meant locating the global IP Address of each of the nodes on the network. After this, unique *hostnames* were assigned for each node which was used alongside the IP addresses, to identify each node.

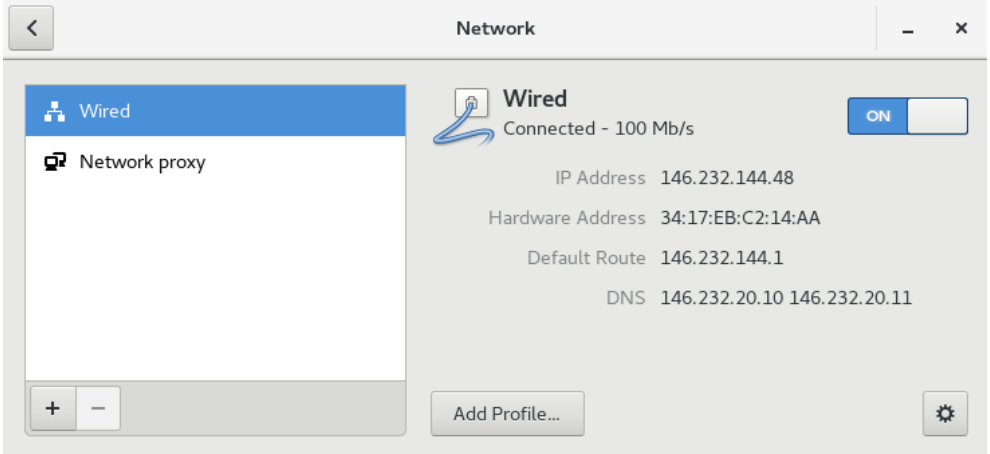


Figure 4.2: The output given when going into the network settings, in order to determine if the computer is connected to the university network.

2. The next step required testing the connection between each of the nodes, to determine if data could be sent and received amongst the nodes using a *ping* command.

3. Given that there was a connection between the nodes, a secure and password-less connection had then to be established. Using SSH (Secure Shell) technology, for each node, an encryption key was generated and shared amongst the nodes. The directory where the keys were stored was then given the appropriate read-and-write permissions.

4. The final step involved testing the secure connection by remotely accessing each node from the NameNode, by using a *ssh* command. A successful and secure connection now meant data could be sent and received without the need for authorisation between the nodes.

The nodes can also be connected to each other by creating a local network. The slave nodes can be directly connected to the master node using a local router or switch (a device that sends data from the internet to computers, or data from computer to computer) which distributes the data among the nodes. Considerations would also then have to be taken on the bandwidth of the router or switch

being used, similar to a network. If a local connection is made, the above steps still have to be followed.

After performing the hardware selection and connecting the nodes together, the data storage and data analytics software could be installed, to create the complete Big Data Analytics system, discussed in sections A.2 and A.3.

4.2 System software

The software required to build the Big Data Analytics Demonstrator, from the operating system to the visualisation software used, is covered in this section. Figure 4.3 provides a visual representation of the different components that are used in this projects Big Data Analysis Demonstrator.

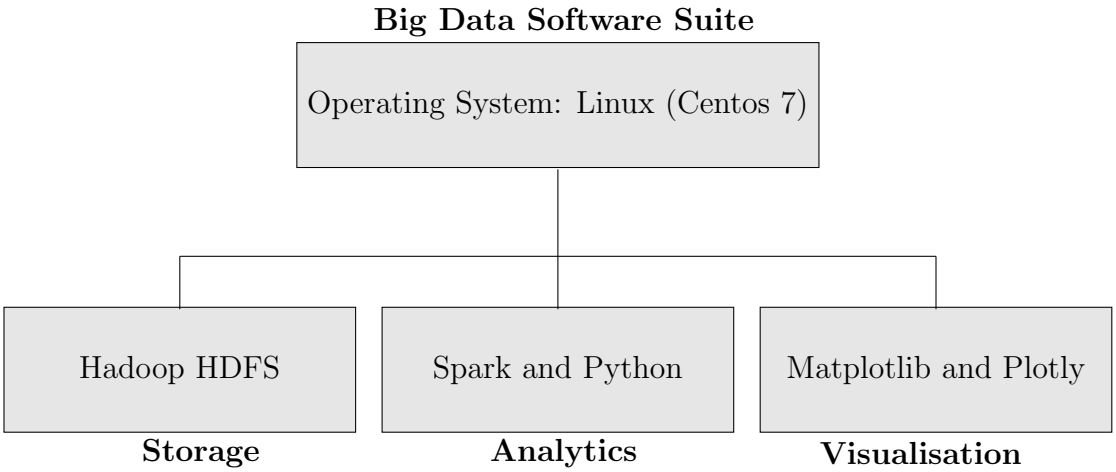


Figure 4.3: A tree of the different software components of the Big Data Analysis Demonstrator, built on top of a Linux OS.

There are many different software solutions that can be configured together to provide a Big Data Analytics solution, similar to the one configured for this project. The reasons for using a certain software is given within each topic covered in this section. At the time of this project, the software used for the demonstrator was deemed appropriate for the intended purpose, scope and goal. With the progress made in technology, software and analytic solutions, this document does not prescribe or make recommendations to use the same configuration, as it will

be out of date at some time in the future. The software to build the demonstrator was chosen because it had to be open-source software (free to use) as set out by the scope of this project and needed to provide the necessary tools to conduct BDA.

4.2.1 Operating system

The operating system (OS) is the initial layer on top of which any software solution is built. The arguments for using the operating system which this Big Data Analytics Demonstrator system uses, are provided. Most operating systems available, from *Windows*, *MacOS* and *Linux* can be used for Big Data Analytics in some form, but usage of the different OSs depends on the specific application; and how the requirements meet the four Vs that define Big Data. It is for this reason, that choosing the specific OS is done by investigating the opinions of persons in the field of Big Data (online opinions/blogs *etc.*) and not a factual basis to choose an OS.

From reading blog posts and articles from the following sources: Quora (2015), Taft (2013), Granville (2014), Schratz (2018), Quora (2016), Bridgwater (2013) and Jain (2015), the researcher concluded that using *Linux* was the better OS choice, as it is used in Big Data Analytics systems and is freely available (open-source). This is in line with the scope of this project which requires the system to be built using open-source software. The specific distribution of Linux, called CENTOS 7 (**version: 7.4.1708**), was chosen after experimenting with pre-built Big Data Analytic suites from Hortonworks and Cloudera, which contain all the Big Data software typically used. These suites, would typically include software such as Hadoop's HDFS, Cassandra, NoSQL *etc.* used to conduct analysis. These suites however, have limited capability for the free-to-use versions, and were therefore not chosen to be used. These suites were used during the initial experimentation and research stage of this project, for the researcher to familiarise and compare commercial Big Data environments and software used in these environments. These pre-built Big Data Analytic solutions were built atop the CENTOS 7 operating system.

After choosing CENTOS and configuring the OS on all nodes, the nodes needed to be connected to each other on the network. This process is previously discussed further in Appendix A in section A.1 and A.1.1. This process was vital to the development of the Big Data Analytics system. The next section outlines the storage system used in the BDAD, along with the steps to configure the system.

4.2.2 Data storage solution

A large component of any Big Data Analytics system is the ability to store a large amount of data and to then access it quickly for analysis. In this section, the installation of the storage solution used in this project is discussed, after outlining why it was chosen.

All of the various Hadoop projects available are open-source Big Data storage solutions, but because the goal of this project is to simply store (and distribute) and analyse data, the HDFS project and format was deemed appropriate. Other projects such as HBase, which focuses on storing large volumes of tabular data, and *Cassandra* which is built to provide high-speed access to data for use in online and near real-time analysis, is not required for this project. For this storage system, the HDFS project, as given in section 2.4.1 was used, because it provided the means to store any data type using the HDFS format. Using the HDFS format also allows for the storage of data to be done in a distributed manner, which allows for fault-tolerance and scalability, as required by the scope of this project.

Detailed steps that were followed to create the Hadoop HDFS environment for this project are discussed further in Appendix A, section A.2. The steps outlined below are all conducted on the NameNode. An overview of the steps to install Hadoop HDFS environment are:

1. The first steps involves creating folders (directories) where the different NameNode, DataNode and Auxiliary information would be stored, and then providing appropriate permission for the Hadoop software to access these folders.

2. After this, the Hadoop HDFS software could then be downloaded and installed, where relevant paths to program files were provided. This allowed the Hadoop software to access relevant on-board software such as Java, essential to the operation. The paths were then located and added in the *.bashrc* location within Linux.

3. The configuration files were then added that the Hadoop installation, which are also provided in section A.2.1, specifically for this three node project configuration. These configuration files can be modified accordingly, if more nodes are to be added. The configuration files specify hostnames of the NameNode, URLs to access different information of the nodes, the number of times data is replicated (set to the number of DataNodes), the block size (as default set to 128 MB) and the location of the configured folders. These configuration files can be changed as more nodes are added. This ability to quickly add computing nodes to the BDA system provides analysts, businesses and organisations the ability to increase storage and computing abilities as the queries become more complex and data sizes increase over time.

4. Next, after modifying the configuration files, the NamNode was formatted and the configuration was saved (however, block size and replication count can be modified if more nodes are added after without loss of current stored data).

5. The final step before starting the HDFS environment was to copy and send all the Hadoop installation files and settings to the other DataNodes. This was done using the *scp* copying command. Because a secure and password-less connection had been previously established, this occurred automatically.

6. Finally, the HDFS environment was started. The dashboards shown in Appendix A.2.2, indicated a successful installation. The different dashboards provide the user or analyst with information regarding the state of the Hadoop environment, from the configured capacity to the state of the nodes, and what files are stored in the HDFS file system.

The method by which to load, remove and manipulate the data on the Hadoop system is provided in Appendix A, under section A.2.3.

The data can then be accessed from the Hadoop system in order to conduct an analysis, with the analytics software discussed next, along with the steps that were undergone to add it to the BDAD.

4.2.3 Data analytics solution

For this project, the analytics suite Spark was chosen. Spark, which is developed to run applications in-memory, uses the RDD (Resilient Distributed Datasets) data packet. This offers faster analysis than MapReduce, which Hadoop uses (IntelliPaat, 2016). The Spark analytics solution provides a built-in machine learning library (Spark, 2018), which is to be used in this project to conduct data mining, pattern recognition and make predictions with the data. A detailed overview of the library *MLib* is provided by Meng et al. (2016) around its performance and scalability compared to others, typically used by MapReduce. This means that no third-party library needs to be configured, ensuring a streamlined process in configuring the system, and fewer possible problems in using the machine learning tools, as the library is native to Spark. The Spark environment is also better suited to machine learning of a non-trivial nature as stated in Table 2.15 by Bekker (2017), whereas MapReduce and Hadoop are best suited to large linear batch processing tasks.

Spark, as discussed in section 2.5, also offers various APIs to develop Spark programs in different languages typically used by the scientific and business community (Jeevan, 2015). This increases the accessibility and ease at which to develop programs, without needing to learn Java (which is considered verbose (Prasanna, 2012)), or use a wide array of wrappers (which are not always optimised) to create MapReduce tasks in languages other than Java. Spark is therefore considered to be easier to work with, as stated by Ellingwood (2016).

Because Spark offers a pre-built suite of libraries (to analyse and visualise data), is able to quickly process the data, and is easier to develop programs in and interpret (code readability), it was chosen to be used in this project. The intended application means that Spark is deemed appropriate for this project,

there are no recommendations made in this document that Spark be used exclusively in other Big Data solutions, as the analytics solution chosen should be done according to the intended application. Larger-scale, linear, and trivial batch processing applications which do not require results in a timely manner, can achieve better performance by using MapReduce rather than Spark (Ellingwood, 2016), for example.

The following are the steps that were followed to create the Spark analytics environment, all from the NameNode. A detailed discussion of these steps is provided in Appendix A.3:

1. The first step involved creating directories for the Spark software, where it is to be installed, on the NameNode. The appropriate permissions were then given, after which the Spark software could be downloaded and installed in the correct folder.

2. Similar to Hadoop, the paths in the *.bashrc* folder had to be added. This includes the location of the Spark software and where to find the executable files for the running of Spark.

3. The single configuration file for Spark had then to be modified, within this file, specifying the Java installation location, giving the correct locations of these files. The directory where Hadoop is located needed to be specified, and finally the number of CPU cores allocated to Spark analytics.

4. The configuration files, the Spark software and directories were then copied to all the DataNodes, similar to Hadoop, using the *scp* command. To add additional nodes to the Spark cluster, the files, directories and software had to be copied to the new node, again using the same *scp* command, shown in section A.3.

5. The final step was to start the Spark environment. A successful installation and configuration of the Spark environment was then shown by the dashboards, in Appendix A, section A.3.1.

For any Big Data Analytics system, a storage system is required in order to provide a point from which to access the data when required for analysis. This

data store, Hadoop's HDFS, was created in the previous section 4.2.2. In order to then access the data stored on the Hadoop system, for analysis by the Spark software, the detailed steps for doing so, are provided in Appendix A, under section A.3.2. Within these steps, are the necessary commands that are used to start and then stop a Spark query within a file, as well as how to submit the file with the query to the cluster. The dashboards to monitor a Spark analytics process being conducted on the cluster are provided in section A.3.1.

The next section focuses on the programming language chosen by the researcher to develop the programs which were used in the analysis of the large datasets, leveraging Spark's built-in APIs.

4.2.4 Programming language for BDA

There are a vast number of programming languages available, all with various qualities and optimised for different applications. The choice of which programming language to use for the BDAD was made after establishing Spark was to be used as the analytics suite. As shown in section 2.5, Spark offers APIs to develop programs in *R, Python, Java* and *Scala*. After examining different online resources Mustatea (2016), Jeevan (2015), Gleeson (2017) and Marr (2015), and concluding that a language selected from among these four would need to be easily readable, have wide support and extensive libraries, Python was chosen. Python is extensively supported in Spark, is less verbose than Java and Scala, as well as being easier to read and learn (making it more accessible for others to understand) and provides a large varying library for machine learning and visualisation. The other languages can also be used as there no recommendation is made to use Python exclusively, but due to the ease with which to learn and interpret Python, as well as the support and extensive libraries, made it more suitable for this project.

4.2.5 Data visualisation solution

For this project, as set out by the project scope, freely available open-source technology is to be used in order to develop a BDAD. This includes different software components, as outlined previously, like Hadoop and Spark to store and

analyse data. The same requirements applied for the visualisation of the data. This is an important component of any analysis, as new insights and results need to be communicated to the end-user(s).

For this project, the *matplotlib* and *plotly* software libraries were used to visualise the results from the analysis. These software packages provide a user with the ability to visualise data through different graphs (*line, scatter, bar, pie etc.*) or geographical maps, and tables to summarise certain results. Other visualisation software such as *Tableau* or *JavaScript* which are also available to visualise the results of an analysis, were not chosen as *matplotlib* and *plotly* provided the tools required to create the desired visualisations. The other software can be chosen if desired, as no recommendations are made to use *matplotlib* and *plotly* exclusively.

4.3 System Design conclusion

In this chapter, the Big Data Analytics Demonstrator components were discussed in detail. The first section looked at the hardware that was used. The goal was to make use of available commodity hardware to develop the demonstrator. After acquiring the three computing nodes they were connected together on the university network to form a computing cluster. These three nodes perform the storage and processing of the Big Data Analytics Demonstrator. The steps required to establish a remote connection between the nodes after connecting them to the network were also provided.

Next, the different software packages and programs were discussed. The first was the storage solution provided. Hadoop's HDFS was deemed appropriate as it provided a means to store and distribute the data on the nodes in a fault-tolerant manner, in line with the project scope. The analytics solution for this project was chosen to be Spark, as this software provided built-in libraries for machine learning, it provides APIs for a variety of languages, is proven to be faster at conducting analysis, and is easier to learn and use than MapReduce (queries do not have to be submitted as separate Map and Reduce jobs in Spark). The language chosen to be used in this project to develop the Big Data Analysis programs was Python. The API, along with strong support for Python is provided with

Spark, making the learning curve, development and interpretation of a Big Data program easier. This allows for a smaller barrier to entry for others interested in understanding the Big Data system.

The system that is designed using this configuration is deemed appropriate and in line with the project scope and goals, along with the proposed architecture in Chapter 3. This is because the Big Data Analytics system provides a means to store and convert the data to HDFS, query the data using Spark, and then use Python's built-in packages *matplotlib* alongside *plotly*, to visualise the insights gained from the analysis. This system that has been developed thus validates the architecture proposed as a possible Big Data Analytics solution. The BDAD model that had been developed therefore fulfils Objective III, where a tool had been developed which could aid analysts in decision-making by analysing Big Data using this demonstrator to perform BDA.

Next, the application of the Big Data Analytics Demonstrator that has been designed and developed, which uses these systems to analyse different structured datasets of varying sizes, is discussed. The analysis will make use of the different libraries (machine learning, visualisation *etc.*) provided by Spark and Python. The goal is to provide a demonstration of Big Data and Big Data Analytics, and the system (BDAD) that had been developed, by comparing it to a system which uses non-Big Data technologies.

Chapter 5

Big Data Analytics Demonstrator Experiments

In the previous chapter, the BDAD was designed and developed, using open-source software, namely Hadoop and Spark, to be able to store data in a distributed, fault-tolerant manner and analyse data in parallel, making use of multiple computing and storage nodes. The chapter discussed how the computing nodes were connected together, how the two software solutions were configured, how datasets could be managed, and queries submitted to the cluster of nodes, which is outlined in Appendix A.

In this chapter, the analytic model, the BDAD system, is validated. Along with this, a *standard system* Python configuration, using non-Big Data technologies, and libraries for machine learning, is used as a means to compare and validate the Big Data Analytics system. When discussing the term *library* or *libraries* in this document, it is with reference to a collection of pre-compiled routines (modules) that are stored and accessed to be used by a program (Beal, 2018). An example of a library is the *PySpark* software, which is a collection of routines or modules allowing Spark to be used with the Python programming language.

The BDAD was validated against the standard system by making use of machine learning and visualisation tools to analyse and interpret different datasets of varying size. The validation involved the machine learning tools according to Figure 2.10, namely, Regression, Classification, Clustering. For each tool, a

number of ML algorithms (techniques) were chosen, and experiments were run on ascending dataset sizes for each algorithm. The time taken to conduct the experiments was recorded, along with the accuracy of predictions made by the different ML algorithms. Each tool makes use of data with a specific data type, with certain features (numerical, categorical *etc.*) that are best suited to the specific tool. Therefore, the datasets and data types used for classification, were different from those used by clustering or regression.

This chapter is structured in the following manner, firstly, the BDAD and standard systems configurations are outlined, taking into account the specific modules and libraries that were used. The standard system consists of one standalone computing node, using Python libraries to conduct an analysis. Thereafter, the ML techniques that were chosen to demonstrate and validate the BDAD are discussed. For each system (Big Data or Standard) the relevant literature around the specific techniques used (k-means, logistic regression *etc.*) are also provided in Appendix B. These are variants from the literature in section 2.3.3 that both systems implement. The literature is to provide a detailed overview and is therefore additional information to gain a better understanding. After this, the different datasets that were used by the different techniques in the validation of the BDAD follows, looking at attributes that made these datasets suitable to the ML algorithm. Then the analytic process to perform the analysis, is provided. Finally, for each ML algorithm (technique), the results of the analysis are discussed, each in a separate section.

The validation serves to illustrate the capabilities of Big Data using a Big Data Analytics system, and comparing it with a standard configuration, thereby demonstrating Big Data to the surrounding industrial engineering community.

5.1 Big Data Analytics Demonstrator and Standard configuration

The Big Data Analytics system used in this analysis, was developed by the researcher, as discussed in Chapter 4. The system consists of three computing nodes, each used to store and analyse the data in a parallel fashion. The focus of

this chapter is on Spark, the analytics software used for this project. The Spark software contains various *machine learning* (ML) algorithms in its library, suited for different applications and datasets. The Spark software offers techniques for all the tools outlined in Figure 2.10 by which to analyse data (except active learning). The number of techniques is however, limited to a select few per tool (*e.g.* k-NN is not available on Spark), therefore only a select few of the techniques for each tool were chosen for comparison purposes. Of the different techniques available under each tool, the general techniques were chosen (*e.g.* choosing only the *k-means* algorithm and not the *bisecting k-means* algorithm, or *linear regression* and not a *random forest regression*).

Next, the standard systems configuration is considered (used to compare with the BDAD). This includes the hardware, programming language and ML library used by this system. The purpose of this system as stated in the introductory section of this chapter, is to compare the results of this system with those of the Big Data Analytics system, and validate the use case of using a Big Data Analytics system.

5.1.1 The standard analytics system

The following is the configuration of the Standard configuration used. Firstly, for the hardware, a single computational node was used. The specifications of this node were as follows:

- CPU: Intel 4th generation Core i7-4770 with a clock speed of 3.4GHz.
- GPU: Integrated Intel HD Graphics 4600.
- RAM: Non-ECC dual-channel 1600MHz DDR3 SDRAM, with 24 GB of memory.
- Networking: Integrated Intel I217LM Ethernet LAN 10/100/1000; supports optional PCIe 10/100/1000 network card.
- Storage: One 500 GB disk drive (ST500DM0 02-1BD142 SCSI Disk Device) and one *Solid State Drive* (SSD) of 256 GB (SAMSUNG SSD SM841N 2.5 SCSI Disk Device).

This hardware combination is the same as the *master node* used in the Big Data Analytics system. This is because the single master node is used, but without coupling it to the rest of the Big Data cluster. The datasets used by the standard system are also stored locally on the machine and not making use of a cluster. For the software, the Python programming language was used as it was the same language used by the Big Data cluster, in order to reduce possible outside factors influencing the results, possibly related to programming language. For the analytics component, the *Scikit-learn* library was used. This library is not the only library available to provide ML solutions, but it is commonly used and contains all the ML algorithms found in Spark. This library is designed to work on single nodes, and not in a distributed manner, as with Spark.

The appropriate ML algorithms and datasets that were used to test and validate the BDAD, are outlined next. This includes accounting for the size of the dataset and data types of the features in the dataset, under each algorithm. After that the analytic process, namely KDD, that was followed to conduct the ML algorithm experiments is discussed.

5.2 Algorithms used in the analysis of both systems

Table 5.1, 5.2 and 5.3 show the different ML algorithms chosen at the time of this project, available in Spark for classification and clustering (Spark, 2018). These are to be used in this project to compare the Big Data Analytics system with the standard configuration. For each Spark algorithm, the associated Python algorithms were used.

Table 5.1: The available Spark regression algorithms, along with the respective algorithms chosen to be used in the comparison of the two systems.

Algorithm	**Available in Spark**	**Algorithm chosen**
Regression	Linear Regression, Generalised Linear Regression, Logistic Regression, Decision Tree Regression, Random Forest Regression, Gradient-boosted tree Regression, Survival Regression, Isotonic Regression	Linear Regression and Logistic Regression

Table 5.2: The available Spark classification algorithms, along with the respective algorithms chosen to be used in the comparison of the two systems.

Algorithm	**Available in Spark**	**Algorithm chosen**
Decision Trees	Decision Tree, Gradient-boosted Tree	Decision Tree
Neural Networks	Multilayer Perceptron Classifier	The Multilayer Perceptron Classifier
Bayes	Naive Bayes	Naive Bayes
Support Vector Machines	Linear Support Vector Machines	Linear Support Vector Machines

Table 5.3: The available Spark clustering algorithms, along with the respective algorithms chosen to be used in the comparison of the two systems.

Algorithm	**Available in Spark**	**Algorithm chosen**
Clustering	k-means, Latent Dirichlet Allocation, Bisecting k-means and Gaussian Mixture Model	k-means

As shown in Tables 5.1, 5.2 and 5.3, there are multiple choices available within each tool, however the general algorithms for each tool were chosen.

5.3 Datasets used to conduct the experiments on both systems

The following table outlines the different datasets that were used to test and compare the performance of the BDAD to that of the standard system. The datasets were freely available datasets that were chosen due to their size (number of rows and number of MB) and data type(s) of the features. Each dataset was therefore suited for specific techniques. The goal was to choose datasets that were large enough to test the BDAD capabilities, while also satisfying the requirements of the ML algorithms.

The datasets where chosen due to their size, as a large enough dataset had to be found which could be used to test and compare the two analytics systems. The next requirement was ensuring the datasets had appropriate data types such that they could be used to conduct experiments on the applicable ML algorithms for those specific data types. This is because not all types of data can be used in all ML algorithms, *e.g.* categorical features are more appropriate for classification, whereas unlabelled data is suited for clustering. The KDD analytics process used is outlined next, given the appropriate datasets and algorithms were chosen.

5.4 Analytics process used conducting the experiments

For this project, the KDD process was chosen due to its generic nature, as concluded in section 2.9. This process could be applied and modified as required, whilst still providing basic guidelines as to how the analytics process should be conducted, to obtain meaningful results. The SEMMA process was not considered as it had limited applications in the SAS software, and CRISP was another implementation of KDD. The KDD process is as follows for this projects analysis:

1. The first step was to obtain an understanding of the application domain: Due to the project requiring a demonstration of BDA, the application domain does not apply. As different datasets were gathered, each having

Table 5.4: A table of the different datasets that were used to analyse the ML algorithms and test the BDAD *vs* the standard system, along with features that made them applicable.

Dataset	Algorithm	Description
NYC Yellow Taxi	Linear Regression, Decision Trees, Multilayer Perceptron Classifier and Naive Bayes	Consists of 28 million trips (each row entry a registered trip) between January and March 2017. Contains numerical continuous, categorical features. Source: NYC Yellow Taxi
Indian Pregnancy Survey	Multinomial Logistic Regression and Linear SVM	Contains numerical, categorical features, of which 21 were used to predict the outcome of a woman's pregnancy. The dataset contains 7.2 million survey entries. Source: Indian Pregnancy Survey
World-wide radiation readings	k-means	Contains 80 million radiation readings of cesium-137, a common isotope found in weapons and reactor accidents. Three numerical, continuous features were used to perform clustering on the locations and intensity of the radiation readings. Source: Radiation Data

different application domains, an analysis would be conducted in multiple fields (domains).

2. The second step involves obtaining a target dataset: For this project however, different test datasets were acquired in order to demonstrate the BDAD ability in analysing different data types, using different machine learning tools and techniques. Therefore, for each tool, a relevant test dataset was used and is discussed under each tool.

3. The third step involves the data cleaning and pre-processing: This step was

applied to all the datasets used in the different tools and techniques, as each dataset required varying amounts of pre-processing to extract the relevant features to then be used by the ML algorithms.

4. Step four involves reduction and projection of data: This could be used to reduce the number of features, if required, to then be used by the ML algorithm. This step was applied as required, depending on the number of features analysed for each individual dataset used. To perform dimensionality reduction, the built-in PCA technique was used.

5. In the fifth step, the goals of the analysis were matched to a relevant ML algorithm: For this project, the ML algorithms are matched with the relevant test datasets acquired, then compared the performance of ML algorithms of the BDAD to the standard system.

Steps (6 – 9): After the completion of the first five steps for each of the datasets, the next steps (6 – 9) of the KDD process were repeated for each ML algorithm. For each ML algorithm, an experiment was conducted for multiple samples and sample sets. Therefore, firstly the algorithm was chosen (6), then the data mining step was applied for each of the experiments that were conducted (7). After this, the results from the experiments were interpreted (8), and finally, a conclusion was given of the overall results for the BDAD and standard system (9).

Before discussing the results from Steps (6 – 9) for the different ML algorithms, the next section considers the process by which results were gathered during the analysis. This includes dividing the datasets into appropriate samples, the relevant results gathered and the notation used throughout the discussion for each of the experiments.

5.4.1 Results gathered from each of the experiments

The analyses of these datasets were divided in the following manner:

Given a dataset X, sample $i \in \{0, 1, ..., 100\}$ was randomly generated for each individual sample set $j \in \{1000; 5000; 50\,000; 500000; 1000\,000; 5000\,000; 10\,000\,000; 20\,000\,000\}$ or $j \in \{1000; 5000; 50000; 500\,000; 1000\,000; 2000\,000;$

4000 000; 6000 000}. The sets j were chosen dependent on the number of rows in the datasets. For each sample i, the sample dataset was divided into a *training set* and *testing set*, where the ML algorithm would be trained on the first and the predictive accuracy would be tested using the *test set*. The samples were divided 70% for the *training set* and 30% for the *test set*. The sample i was then analysed for sample set j, whereby the mean percentage accuracy (μ) and standard deviation (σ) values in predicting the outcome for all samples i of sample set j was taken, along with the time taken per sample (measuring the mean and standard deviation time) for the BDAD and standard system respectively. Thus, for example, sample set i_{1000} had a total of one hundred randomly generated samples each of which had 1000 rows, which were divided into a training and test set (700 and 300 rows respectively), each analysed, and then stored the results.

Thus, the following information was collected for the BDAD and standard system (BD, SS), using the notation given in Table 5.5: The notation BD indicates

Table 5.5: Results gathered from the analysis of the BDAD and standard systems with the notation used in this project to denote the results.

Results Gathered	**Notation used for the results from the BDAD and Standard System**
Mean Percentage Accuracy of the Predictions	$\mu_{\%,\text{BD}}$ and $\mu_{\%,\text{SS}}$
Standard Deviation of the predictions	$\sigma_{\%,\text{BD}}$ and $\sigma_{\%,\text{SS}}$
Mean Time per analysis	$\mu_{t,\text{BD}}$ and $\mu_{t,\text{SS}}$
Standard Deviation of the time per analysis	$\sigma_{t,\text{BD}}$ and $\sigma_{t,\text{SS}}$

the BDAD system as Big Data (BD), and the standard system is indicated by SS (Non-Big Data).

Given the notation to represent the different results collected, and the method by which the results were gathered, the results from the experiments for each ML algorithm are now considered, continuing with the steps 6 – 9 of the KDD analytics process outlined previously. The results from the regression experiments are

discussed first followed by the different classification algorithms, and finally the results from the clustering experiments. After this, an overall conclusion to the experiments is provided. To provide additional background on the specialisations and variants of the ML algorithms used by Spark (2018) and Scikit-learn (2017) in these experiments, a literature review can be found in Appendix B.

5.5 The regression experiments

Regression is typically applied to datasets where the relationship between a dependent variable is to be examined against an independent variable(s) (Alpaydin, 2010). By examining this relationship, a trend is drawn by which predictions can be made. For this project, linear and logistic regression were applied to the specified datasets. As discussed, the time taken to conduct a Regression analysis and the accuracy of the analysis is to be compared between the BDAD and Standard configuration. Before this, a more detailed look at the data types required when conducting a regression analysis is provided, as not all data types are applicable to a regression analysis.

5.5.1 Data types required for regression

Regression as outlined examines the relationship between a dependent and independent variable(s), it is therefore a form of *supervised* ML as it uses labelled data. Using Campbell and Campbell (2008), the initial step is to determine if the data is qualitative or quantitative. The data, if qualitative cannot be used by a regression function as it groups multiple occurrences within a single value (*e.g* a collection of people owning an iPhone = 1 or Samsung = 0). Using this example, if there was data on how long the cellphone lasted for each individual, given an iPhone or Samsung, this relationship could be examined using regression. This is a form of quantitative data that is continuous (time it takes to charge a cellphone battery).

5.5.2 Logistic regression

The literature provided in section 2.3.3.2 regarding logistic regression provided the basis of the algorithm's function, of which a detailed overview on the modified version for Spark and Scikit-learn is provided in Appendix B (sections B.1.1 and B.1.2 respectively). Because the function is able to classify the results given the logistic function, it can also be regarded as a classification algorithm Spark (2018) and Scikit-learn (2017). The Indian Pregnancy dataset was used in this analysis, as the algorithm required the 'outcomes' or classification itself to be binary (0 and 1). The dataset is however able to make these binary predictions using multiple features.

Table 5.6: The results from conducting logistic regression with the BDAD and standard system, taking the mean results for the 100 samples of the different sample sets.

	Spark				Scikit-learn			
Sample Set	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation
1000	98.4846	0.01230	2.5302	0.9710	99.3867	0.0035	0.0083	0.00168
5000	99.6115	0.00124	3.2309	1.1038	99.8920	0.00058	0.0242	0.0018
50 000	99.6422	0.00041	6.9813	1.3917	99.9999	9.3804e-06	0.2902	0.0736
500 000	99.6430	9.7573e-05	41.9409	6.4416	99.9999	1.9175e-06	3.6279	0.8801
1 000 000	99.6450	8.9492e-05	75.085	12.0108	99.9999	1.2866e-06	7.6268	1.9418
2 000 000	99.6455	5.0433e-05	143.176	24.8511	99.9999	8.2334e-07	16.2777	6.1837
4 000 000	99.6550	3.4931e-05	278.9078	54.4175	No Result	No Result	No Result	No Result
6 000 000	99.6566	2.6019e-05	409.5056	83.5848	No Result	No Result	No Result	No Result
Mean of Mean	99.4979	0.0019	120.1697	23.0965	99.8797	0.0007	4.6425	1.5138

Table 5.6 provides the results from conducting a logistic regression analysis on the BDAD (Spark) and the standard system (Scikit-learn). The dataset contained a total of 7.2 million survey entries. The goal of the logistic regression was to classify given a woman is pregnant, if the child is to be born alive (1) or still/abortion (0). A total of 21 features were used to conduct this classification. The features-set were the following: *Age, Sex, Marital Status, Religion, State, District, Smoke, Chew tobacco, Drink alcohol, Health Insurance, Disability status, Treatments for any injuries, Have illness, Currently Attending School, Education Level, Currently Pregnant, Previous Pregnancies, Age of Previous Pregnancy,*

Months Pregnant, Currently Menstruating, Rural or Urban Living. After conducting pre-processing, this included removal of null values, and making relevant assumptions, the logistic function could be run on a maximum of 6 million rows (people). Some of the assumptions were made by researching smoking statistics amongst women in India, usage of contraceptives, income and literacy levels in order to fill in missing values for the features that were used. By simply removing all null values, the dataset contained too few entries, therefore assumptions had to be made.

The results in Table 5.6 show that the standard system performed the classification faster and with more accuracy than the BDAD when working with the smaller sample sets. For both systems the standard deviation $\sigma_{\%,\mathrm{BD}}$ and $\sigma_{\%,\mathrm{SS}}$ from the mean accuracies was low, 0.0019% and 0.0007% respectively. This indicates there were small shifts in predictions from the mean. Taking the global mean for all sample sets for the BDAD and standard system, the difference was 0.3818% between the systems. As shown, the accuracy also increased for the BDAD as sample sets were increased, whereas the standard system was unable to conduct the logistic regression analysis for sample sets above 4 000 000 rows. This means the BDAD was able to analyse more data with increasing accuracy. It has to be noted however, that the time taken to conduct the analysis did increase more significantly compared to the standard system as sample sets were increased. This is due to the network overhead from the parallel computations, as the nodes were connected over a network, increasing latency, compared to the standard system which performed the analysis locally.

This analysis demonstrates the ability of the BDAD over the standard system. As the sample sets were increased, the BDAD was able to perform the analysis with good accuracy, whilst also being able to analyse the largest sample set compared to the standard system. The standard system's accuracy was greater than the BDAD when working with small sample sets, with a relatively low percentage difference. A comparison could not be made between the two systems for the largest sample sets, therefore it can be assumed (looking at the trend) that if the dataset were to be larger, the BDAD would perform more accurate predictions. This also shows the superiority of the BDAD in analysing larger

datasets compared to the standard system; however the time taken for a larger analysis would increase as well.

5.5.3 Linear regression

Linear regression (LR) is applied to instances where the relationship between the dependent and independent variable is linear, as the value of the independent variable is altered, the dependent variable increases or decreases linearly in relation to the independent variable. This was given by

$$\phi(x) = \beta_0 + \beta_1 x \tag{5.1}$$

in Chapter 2 under the supervised ML section. How Spark and Scikit-learn expand and apply this concept, can be found in Appendix B in sections B.1.3 and B.1.4 respectively.

The dataset used to conduct the linear regression analysis on both systems was the NYC Yellow Taxi dataset. This dataset contains 28 million data entries, each entry represented a registered yellow cab taxi trip. The trips were between January and March 2017. For this analysis, the cost of one trip *fare amount* in USD was predicted, given the distance travelled (*trip distance*) in miles, and the time taken per trip (*trip duration*) measured in seconds. The reason for using both features to predict the cost of a trip was because the NYC Yellow Taxis take into account the time of a trip (given a taxi is stationary for a period of time) and the distance travelled (when the taxi is not stationary). Due to the size of the dataset, it was decided to remove any outliers from the dataset as part of the pre-processing. This left the total number of complete recorded trips to be 21 million entries (rows). Using this pre-processed dataset, the linear regression analysis was performed. As previously stated, this was done on increasing sample sets, for each sample set 100 randomly chosen trips were selected and analysed.

The results from the linear regression analysis are provided in Table 5.7. The results indicate the standard system providing a high percentage accuracy in the prediction of the trip cost $\mu_{\%,\mathrm{SS}} = 94.3220\%$ compared to the BDAD $\mu_{\%,\mathrm{BD}} = 85.71\%$ for the global means. The difference between the standard system and BDAD was 8.612%. The accuracies for both systems also decreased on average

Table 5.7: The results after conducting linear regression with the BDAD and standard system, taking the mean results for the 100 samples of the different sample sets.

	Spark				Scikit-learn			
Sample Set	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation
1000	86.4391	0.0801	0.1433	0.0349	95.0238	0.0472	0.0094	0.0703
5000	86.1597	0.0358	0.1461	0.0416	94.3360	0.0320	0.0112	0.0877
50 000	85.6034	0.0168	0.1902	0.0998	94.3479	0.0166	0.0097	0.0620
500 000	85.6380	0.0055	0.5654	0.1180	94.1742	0.0055	0.0213	0.0699
1 000 000	85.3789	0.0059	0.9627	0.2234	94.1195	0.0053	0.0438	0.1492
5 000 000	85.5893	0.0018	4.5010	1.2337	94.1305	0.0023	0.1627	0.1121
10 000 000	85.4699	0.0023	9.1153	2.0978	94.1218	0.0019	1.7429	6.2129
20 000 000	85.4013	0.0011	17.5966	3.2027	No Result	No Result	No Result	No Result
Mean of Mean	85.7100	0.0187	4.1526	0.8815	94.3220	0.0158	0.2858	0.9663

as sample sets were increased. When considering the standard deviations, they decreased on average as the sample sets were increased, indicating that even though the accuracy decreased, the values remained relatively close to the mean predictions. For the time and prediction accuracies for both systems, the standard deviation remained relatively low, indicating small variations from the means. Importantly, however, the BDAD was able to analyse larger sample sets, as the standard system was unable to perform the linear regression on the largest sample set of 20 000 000. The time taken to conduct the analysis also increased as the sample sets increased for both systems, but the BDAD experienced a greater increase than the standard system, with global means $\mu_{t,\text{BD}} = 4.1526$ seconds and $\mu_{t,\text{SS}} = 0.2858$ seconds respectively.

The results indicate that the standard system outperforms the BDAD, but that for both systems, the accuracy in predictions did decrease slightly as sample sets were increased. Along with this the results show the ability of the BDAD to analyse larger datasets than the standard system. The BDAD did, however, take longer to conduct an analysis, also increasing more as the sample set was increased compared with the standard system. Therefore, if an analyst is working with smaller datasets, the standard system is more favourable, but as sample sets increase and it is required to analyse larger datasets, the BDAD is the best alternative.

5.6 The classification experiments

Classification is done in order to predict an outcome given a set of decisions or conditions that are met Quinlan (1986). Classification can only be conducted on data that allows for some form of classification or prediction to occur, therefore only certain data types are associated with classification and used by classification algorithms. This is firstly investigated. As outlined in section 5.2, for the classification algorithms, each is briefly discussed for Spark and Scikit-learn, regarding the relevant literature, then the results from the validation phase are discussed, comparing the algorithm's performance for the two systems.

5.6.1 Data types required for classification

As classification falls under supervised learning (Jiawei et al., 2012), the data type required for classification algorithms is labelled data (group of samples given a means of identification through a tag (Stackoverflow, 2013). This could be an image that is provided a tag or an item code given a specific code (*e.g.* Apple = 1, Samsung = 0).

When choosing data for a specific classification algorithm, it should also be considered if the data is qualitative or quantitative. Qualitative is typically categorical data (Apples, Oranges and Pears or Christian, Jewish, Muslim). Quantitative data is numerical, where the data is either discrete (Number of Taxis in a city) or continuous (temperature, GDP) (Campbell and Campbell, 2008). If the data contains labels as strings (the label by which the factors are grouped) these would first have to be converted into categorical number labels to be used by the classification algorithm (*e.g.* Fraudulent = 1, Cleared = 0). Continuous data should be converted to categorical data in order to classify the data according to a given category (Kotsiantis, 2007), though the use of 'bins' (assigns a categorical value, given the continuous value falls within a specified range).

5.6.2 Decision trees

The decision tree (DT) algorithm literature and concept is outlined in section 2.3.3.2 and the specific Spark and Scikit-learn algorithm variants from Spark

(2018) and Scikit-learn (2017) can be found in Appendix B (section B.2.1 and B.2.2).

The NYC Taxi dataset was used to perform a decision tree analysis with the BDAD and standard system. This dataset contains 28 million taxi trips and after pre-processing there were 21 million usable trips for the ML algorithm to use. For this analysis, during the pre-processing, the cost per trip (*fare amount*) class chosen to be classified, was modified from continuous numerical values into categorical 'bins'. This was in order for the ML algorithm to perform classifications, as categorical data is required when making classifications. There were 17 bins identified by the following range: $[0-2]$; $[2-4]$; $[4-6]$; $[6-8]$; $[8-10]$; $[10-40]$; $[40-60]$; $[60-80]$; $[80-100]$; $[100-150]$; $[150-200]$; $[200-250]$; $[250-300]$; $[300-350]$; $[350-400]$; $[400-450]$; $[450-\text{inf}]$. The features that were used to classify the trip cost according to these bins were the *trip distance* (as previously used in the linear regression analysis) as well as the *payment type* (cash, credit card, *etc.*), the *rate code* (*e.g.* denoting if the trip is during a *surcharge* period) and the *number of passengers* for each trip. Given these factors, a cost per trip could be predicted, as the total trip cost can be increased or reduced given these various factors.

Table 5.8: The results from conducting a decision tree analysis with the BDAD and standard system, taking the mean results for the 100 samples of the different sample sets.

	Spark				Scikit-learn			
Sample Set	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation
1000	71.6804	0.0292	0.4891	0.2556	66.6833	2.8762	0.0174	0.0380
5000	74.0358	0.0114	0.6936	0.3045	67.5307	1.4801	0.0171	0.0042
50 000	73.3363	0.0040	2.0904	0.4575	67.9578	0.7871	0.0631	0.0156
500 000	73.7469	0.0012	15.0984	2.8815	68.4453	0.3884	0.6554	0.0599
1 000 000	73.6141	0.0034	29.3641	6.3144	68.5478	0.1420	1.4245	0.0964
5 000 000	73.6631	0.0006	144.6270	29.9247	68.5447	0.0416	8.5576	0.8026
10 000 000	73.6579	0.0006	31.8984	6.8336	68.5418	0.0257	17.9117	1.6441
20 000 000	73.6631	0.0008	577.5007	123.9232	No Result	No Result	No Result	No Result
Mean of Mean	73.4247	0.0064	100.2202	21.3619	68.0359	0.8202	4.0924	0.3802

The results of the decision tree analysis are given in Table 5.8. The results indicate a higher mean accuracy in predictions for the BDAD compared with the

standard system. The BDAD obtained a global mean accuracy (mean accuracy values for all sample sets) of $\mu_{\%,\text{BD}} = 73.4247\%$ compared to $\mu_{\%,\text{SS}} = 68.0359\%$ for the SS. Together with this, the BDAD obtained lower standard deviations as the sample sets were increased, compared with the standard system $\sigma_{\%,\text{BD}} = 0.0064\%$ to $\sigma_{\%,\text{SS}} = 0.8202\%$. The general trend also indicates that as the sample sets were increased, the accuracy in the predictions increased for both systems. Given these results, the global mean time for the BDAD to analyse a single sample was $\mu_{t,\text{BD}} = 100.2202$ seconds with a standard deviation of $\sigma_{t,\text{BD}} = 21.3619$ seconds, for all sample sets was significantly higher than that of the standard system, $\mu_{t,\text{SS}} = 4.0924$ and $\sigma_{t,\text{SS}} = 0.38202$ seconds.

Therefore, the BDAD was able to analyse and predict the trip cost with a higher accuracy as sample sets were increased, with lower standard deviations from the mean. The BDAD did however take longer to conduct the analysis as the sample set increased, but it was able to analyse all sample sets. The standard system was unable to analyse the largest sample set of 20 000 000 rows and therefore no results were obtained. This analysis successfully indicates the use case of the BDAD and its ability to analyse larger datasets compared to the standard system, but with each analysis taking longer compared with the standard system.

5.6.3 Naive Bayes

The Bayes theorem concept and method were previously discussed in section 2.3.3.2. The Bayes theorem attaches conditional probabilities to illustrate a relationship between a set of variables (Kotsiantis, 2007). Certain probabilities can have a dependence on previous variable probabilities, for this reason, naive bayes (NB) applies the theorem, but assumes independence between every pair of features (variables) Scikit-learn (2017). For further reading on the naive bayes implementations by Spark (2018) and (Scikit-learn, 2017) respectively, Appendix B.2.3 can be consulted.

The dataset used to perform the naive bayes classification was also the NYC Yellow Taxi dataset. The same pre-processing was conducted before conducting

the analysis as in the DT analysis. Therefore the outliers were removed, and the trip cost was allocated separated into different bins.

Table 5.9: The results from conducting a naive bayes analysis with the BDAD and standard system, taking the mean results for the 100 samples of the different sample sets.

	Spark				Scikit-learn			
Sample Set	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation
1000	83.4983	0.0258	0.2586	0.2285	64.8100	4.8724	0.0201	0.0456
5000	84.1629	0.0186	0.4266	0.2023	60.2440	4.9056	0.0236	0.0062
50 000	79.4226	0.0758	1.6467	0.3721	55.6719	5.0318	0.1309	0.1236
500 000	63.3118	0.1954	14.6886	2.6953	58.4898	7.9217	1.3260	0.1224
1 000 000	52.8488	0.1802	26.8342	6.0988	62.7130	4.8256	2.8452	0.1520
5 000 000	41.9528	0.0009	135.6457	28.8395	62.8396	2.8657	15.8140	0.4882
10 000 000	41.4895	0.0007	274.2144	61.5116	63.2821	2.3236	32.5449	1.1444
20 000 000	41.3729	0.0008	564.2209	107.6703	No Result	No Result	No Result	No Result
Mean of Mean	61.0075	0.0623	127.2420	25.9523	61.1501	4.6781	7.5292	0.2975

Table 5.9 provides the results from conducting the naive bayes analysis on the BDAD and standard systems. The results indicate the BDAD was able to deliver higher mean prediction accuracies compared with the standard system when working with smaller datasets with smaller standard deviations. The accuracy of the BDAD remained higher for sample sets less than 500 000 rows, where from 1 000 000 rows and greater, a significant drop in predictive accuracy was experienced. These results were contradictory to what the researcher was expecting and warranted further investigation, as the accuracy was expected to increase with larger sample sets. As the accuracy of the predictions was higher with smaller sample sets, the possibility of *overfitting* or *underfitting* was of concern. As stated in Raschka (2014) and Nautiyan (2013), *overfitting* causes the predictions to decrease given too many datapoints and the algorithm fits the noise and variations of the training set, to ultimately not predict the test dataset with good accuracy.

A solution to this as proposed by Nautiyan (2013) is to perform cross-validation, regularisation, pruning or stopping the algorithm sooner. As stated in Spark (2018) the algorithm is optimised for text classification, which could be a possible reason for the poorer performance, but this could not be validated as an appropriate explanation for poorer performance on larger sample sets. This is

because measures to account for possible *underfitting* or *overfitting* as the cause for poor performance, were not tested (prevent possible alteration to the experiments being conducted). Given this, the results do, however, indicate the BDAD providing predictions that are more closely centred around the mean, as the *global mean* for the standard deviation on the BDAD was lower than the standard system, $\sigma_{\%,\text{BD}} = 0.0623\%$ compared to $\sigma_{\%,\text{SS}} = 4.6871\%$. The standard system also performed with 21.7926% higher predictive accuracy, when comparing the largest sample set analysed by the standard system of 10 000 000 rows to the BDAD.

Provided the appropriate measures were taken and datasets used when using the naive bayes algorithm on the BDAD, the BDAD is able to also analyse larger datasets compared with the standard system, as the largest sample set could not be analysed by the standard system.

5.6.4 Linear support vector machines

The goal of Support Vector Machines (SVM) as outlined in Chapter 2, is to develop a plane in a high or infinite dimensional space, where the plane's distance is maximised between training points in order to best classify a point on either side (Tong and Chang, 2001). The literature regarding the Spark (2018) and Scikit-learn (2017) specialisations of the SVM algorithm can be found in Appendix B.2.4 for further background.

The Indian Pregnancy dataset was used in this analysis, as the SVM algorithm for Spark specifically was linear and required the 'outcomes' or classification to be binary (0 and 1). The dataset is however able to make these binary predictions using multiple features. For this reason, the NYC Taxi dataset was not used. As stated in Spark (2018), the algorithm is able to perform binary classification using multiple features, similar to the logistic regression algorithm. Therefore, similar pre-processing steps and assumptions were made before conducting the analysis, also making use of the 21 features to predict if a pregnant woman would deliver a baby *alive* (1), or *still/have abortion* (0). As with logistic regression, after these steps, there were 6 million survey entries (rows) from which to perform linear SVM analysis.

Table 5.10: The results from conducting a SVM analysis with the BDAD and standard system, taking the mean results for the 100 samples of the different sample sets.

	Spark				Scikit-learn			
Sample Set	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation
1000	90.4227	0.0207	1.3632	0.5643	99.9300	0.0024	0.0183	0.0076
5000	92.3589	0.0115	1.4817	0.9078	100.0000	0.0000	0.0508	0.0072
50 000	93.3065	0.0203	5.4798	1.9139	100.0000	0.0000	0.6555	0.0634
500 000	93.3915	0.0189	40.6084	8.9817	100.0000	0.0000	9.1944	2.4336
1 000 000	92.8966	0.0140	75.0180	17.7073	No Result	No Result	No Result	No Result
2 000 000	92.6483	0.0099	145.7644	33.7583	No Result	No Result	No Result	No Result
4 000 000	93.4060	0.0189	318.3078	63.3384	No Result	No Result	No Result	No Result
6 000 000	92.8060	2.5861e-05	457.7310	95.5840	No Result	No Result	No Result	No Result
Mean of Mean	92.6545	0.0143	130.7193	27.8445	99.9825	0.0006	2.4798	0.6280

After performing the linear SVM analysis, the results were collected and provided in Table 5.10. The standard system was able to classify the results with higher accuracy compared to the BDAD when analysing smaller datasets, as the researcher expected. The standard system was able to predict with a mean value of 6.6085% higher than the BDAD. For the largest sample set it could analyse 500 000 records. The results show low standard deviations for both systems for the accuracy in predictions, the global mean values were $\sigma_{\%,\mathrm{BD}} = 0.0143\%$ and $\sigma_{\%,\mathrm{SS}} = 0.0006\%$ respectively. This showed the prediction accuracy remained relatively close to the mean values, with small variations. As the sample sets were increased in size, the standard system was unable to provide results for sample sets above 1 000 000 rows. The time taken for the BDAD to conduct the analysis on larger sample sets, as with other ML algorithms, did increase significantly, with the largest sample set taking 457.7310 seconds per sample i to perform a linear SVM analysis.

The results show that for smaller sample sets the standard system provides predictions with a higher prediction accuracy, in less time than the BDAD. As the sample sets increase however, only the BDAD is able to perform a linear SVM analysis. Provided time per analysis is less of a concern, and modifications to the dataset or algorithm was made to increase the accuracy, the BDAD is a better solution if the dataset or sample set were to be considerably large such that the

standard system is unable to perform the analysis.

5.6.5 Multilayer perceptron classifier

The Multilayer Perceptron Classifier (MPC) is an ANN algorithm based on *feed-forward artificial neural networks*, and uses backpropagation during the learning phase of a model, as stated by Spark (2018). The MPC is the only ANN currently available on Spark, and was therefore chosen to demonstrate the ability of the BDAD to perform classification using an ANN algorithm. ANNs are typically applied in image recognition and artificial intelligence applications as well as text classification. The basic concept of the ANN was discussed in Chapter 2, which uses neurons, with input variables and weights for each variable. The weighting for each variable is adjusted accordingly, and given certain values for the input, it passes a threshold function, resulting in the specific output and classification. A specialisation of the MPC method used by Spark (2018) and Scikit-learn (2017) is provided in Appendix B, in section B.2.5, and can be consulted for further reading. As with the other algorithms, the MPC algorithm was run on the BDAD with different samples and sample sets while comparing the time and accuracy of each analysis to the standard system.

The dataset used to perform the MPC analysis was the NYC Yellow Taxi dataset as used previously with decision trees and naive bayes, where the cost of a trip (*fare amount*) was predicted to fall into the 17 different price categories. This was to ensure the algorithm could classify categorical data and not continuous data. The amount of data also remained the same, after pre-processing there were 21 million taxi trip datapoints from which to classify. The features used in the classification also remained the same, with the *trip distance*, *number of passengers*, *payment method* and *rate code* (code to denote for example, if the trip is currently incurring a *surcharge*) used to classify and predict the appropriate price bin for each trip.

The results after the MPC analysis with both the BDAD and standard system are shown in Table 5.11. From the results, the mean of mean for the prediction accuracy of the BDAD is higher than that of the standard system, with the BDAD $\mu_{\%,\mathrm{BD}} = 73.7873\%$ and the standard system $\mu_{\%,\mathrm{SS}} = 67.5662\%$. This

Table 5.11: The results from conducting the multilayer perceptron classifier analysis with the BDAD and standard system, taking the mean results for the 100 samples of the different sample sets.

	Spark				Scikit-learn			
Sample Set	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation	Accuracy (mean %)	Standard Deviation	Time (seconds)	Standard Deviation
1000	72.4399	2.4963	2.4517	0.7913	66.7700	2.8311	0.1962	0.0649
5000	74.6013	1.1171	3.9757	0.9532	67.1793	1.6115	0.4312	0.0815
50 000	73.7304	0.4647	17.7136	2.4980	68.6769	1.2807	4.2976	0.4161
500 000	73.8627	0.3230	157.7070	17.4615	67.7510	1.2523	46.0990	6.0626
1 000 000	73.8782	0.1351	288.8610	33.1531	67.4536	1.0190	99.7044	11.3860
5 000 000	73.9682	0.0756	1610.2486	164.7709	No Result	No Result	No Result	No Result
10 000 000	73.8044	0.1232	2494.7107	305.3164	No Result	No Result	No Result	No Result
20 000 000	74.0130	0.0427	5493.6162	522.5580	No Result	No Result	No Result	No Result
Mean of Mean	73.7873	0.5972	1258.6606	130.9378	67.5662	1.5989	30.1457	3.6022

indicates that the BDAD was on average, with a difference of 6.2211%, more accurate in classifying and predicting the trip costs than the standard system. The results also indicate that the accuracy remained relatively similar as the size of sample sets was increased on the BDAD, with the largest sample set having a high accuracy with the lowest standard deviation $\sigma_{\%,BD} = 0.0427\%$. The BDAD did however take significantly longer in making the predictions as the sample sets were increased, with the largest sample set 20 000 000 rows taking 5493.6162 seconds per sample. The mean of mean $\mu_{t,\text{BD}} = 1258.6606$ seconds for the BDAD, compared to the standard system $\mu_{t,SS} = 30.1457$ seconds per sample. This is a significant time difference, but the standard system was unable to perform the MPC analysis on sample sets larger than 1 000 000 rows. This therefore indicates the point at which the BDAD is more favourable as an analytics system, as the BDAD was able to analyse the largest sample sets, with higher accuracies for all sample sets, compared with the standard system.

Given that longer times to conduct the analysis are of less importance, the MPC algorithm on the BDAD will provide more accurate predictions in the classification results, with small standard deviations when working with larger amounts of data. If time is more of a concern, and the datasets are smaller, and predictions on average 6.2211% less than the BDAD are not significant given the application, the standard system could be used.

5.7 The clustering experiments

The final tool used for Big Data Analysis was *unsupervised learning*, commonly referred to as *clustering*. The technique used was *k-means*, the goal of this technique is to take data that has no label, and perform a similarity analysis by grouping common data points together. A criterion is then given to determine how well the grouping has been done and what constitutes a good grouping or cluster (Hansen and Jaumard, 1997). Some of the applications for clustering are text-mining, pattern recognition, image processing and web engines (Bijuraj, 2013). The *k-means* clustering method is outlined as used by Spark and Scikit-learn, after discussing the appropriate data types required for clustering. Similar to the previous validations outlined, the clustering time and accuracy of both the Big Data and Standard systems were recorded and compared.

5.7.1 Data types required for clustering

The data type as discussed by Witten et al. (2016) required when conducting any cluster analysis is to make use of data that is un-labelled. For example, when clustering bank transactions to determine fraudulent behaviour, given a 'normal' cluster of data points which represent previously similar transaction amounts by a consumer, once a payment is added that does not fall into the 'normal' cluster of transaction amounts, that specific data point is grouped into an 'abnormal' cluster (Bolton and J, 2018). If the data points were to be given date labels, then the data points could not be used for clustering, but rather for performing regression analysis, predicting future transaction amounts to be made.

Nominal, numeric and ordinal data are all data types that clustering algorithms can use to form groups or clusters. Another consideration when working with different data types is to apply feature scaling. Feature scaling is applied to data if the clusters would be biased towards a given feature(s), especially if the features have attributes which vary in size (Mohamad and Usman, 2013). An example of this would be to cluster data of different measures, height (cm) with mass (kg). As stated by Mohamad and Usman (2013), feature scaling should be applied to data during pre-processing before conducting clustering algorithms which use Euclidean distance to calculate clusters.

5.7.2 k-means

The clustering algorithms as discussed in the literature review included distance-based clustering methods as well as hierarchical methods. For this project, the commonly used k-means distance-based method was used to demonstrate the Big Data Analytics system's capability in analysing un-labelled data. The literature from Spark (2018) and Scikit-learn (2017) on their respective implementations of the k-means clustering techniques is outlined in Appendix B.3.

The validation of the clusters is done by evaluating an elbow plot of the Sum Square Error (SSE) values, for the different clusters that have formed. The SSE function was discussed in section 2.3.3.3. This metric measures the error in the distance between each data point and the cluster centre, the cluster number where there is a significant equalisation (difference in following SSE values decreases significantly) in the SSE values, or 'elbow', is considered the appropriate number of clusters for the analysis Lab (2013). The resulting clusters have then to be evaluated by the analyst to determine if the clusters are appropriate within the context of the data. This evaluation method had to be used due to the un-labelled nature of the data.

As previously discussed, to evaluate the unsupervised ML abilities of the BDAD compared to the standard analytics system, the Radiation dataset was used. This dataset was chosen due to its large volume of records (rows of data), which were un-labelled. This made it ideal for Clustering. The dataset had to firstly be filtered and pre-processed, removing any null values and selecting only the radiation recordings where 'cpm' or the cesium readings associated with nuclear rectors and reactor explosions (Wessells, 2012), were selected. This left 79 million rows (readings) to be analysed. The relevant fields selected for the clustering from the radiation datasets was the location namely *Latitude*, *Longitude* and the cpm *Value*. The k-means clustering algorithm then clustered the different locations where the readings were taken.

For the initial analysis on the BDAD, assumptions had to be made about the number of clusters, as required by the k-means algorithm. Due to the un-labelled nature of the data, clusters have to be chosen given the context of the data and query. The researcher therefore chose to analyse the radiation readings

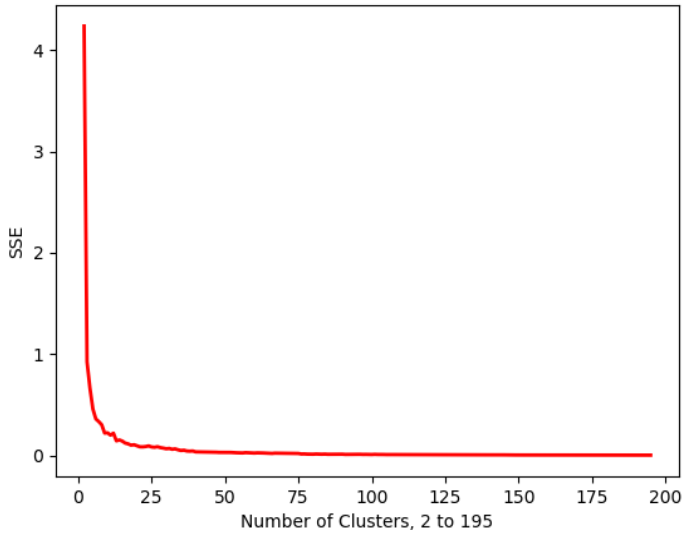


Figure 5.1: The elbow plot of the 195 clusters, after performing k-means, taking the SSE values, from the BDAD, for the radiation dataset.

according to country (as the dataset contains world-wide readings), to inspect possible radiations readings per country. As of the time of this project, there are 195 countries (Stratfor, 2018), and therefore 195 clusters were chosen.

When considering the elbow plot for the initial cluster analysis, shown in Figure 5.1, the 195 clusters were not considered the best number of clusters. Another analysis was conducted with five clusters. The elbow plot given in Figure 5.2 where three clusters are shown, shows that three clusters are considered best. The SSE value at three clusters shows the initial 'elbow' in the plot which is considered best. Considering the data represents worldwide radiations measurements, the results from clustering the data into three clusters shown in Figure 5.3, do not yield enough meaningful insight regarding readings, which an analyst could use (merely three large clusters divided across a third of the earth). When clustering the data according to the 195 countries, closer, a more densely packed number of clusters had formed. The resulting Figure 5.4, is of 400 000 readings that were visualised, as the system was unable to visualise the entire dataset. These clusters, when considering the US as an example, roughly represent each US state (not what the researcher originally predicted would occur). The results could therefore be used to determine radiation intensities according to each state.

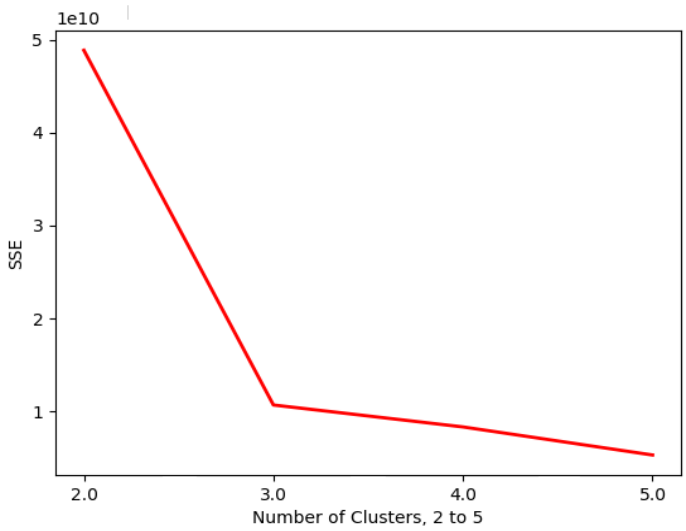


Figure 5.2: The elbow plot of the five clusters, after performing k-means, taking the SSE values, from the BDAD.

Similarly, when looking at Europe, the clusters have formed roughly according to country, allowing for radiation readings to be inspected per country. This is not considered a good result as given by the elbow plot, and the 195 clusters have not divided according to every country, but the results do however indicate that using more than three clusters yields greater insights into the data.

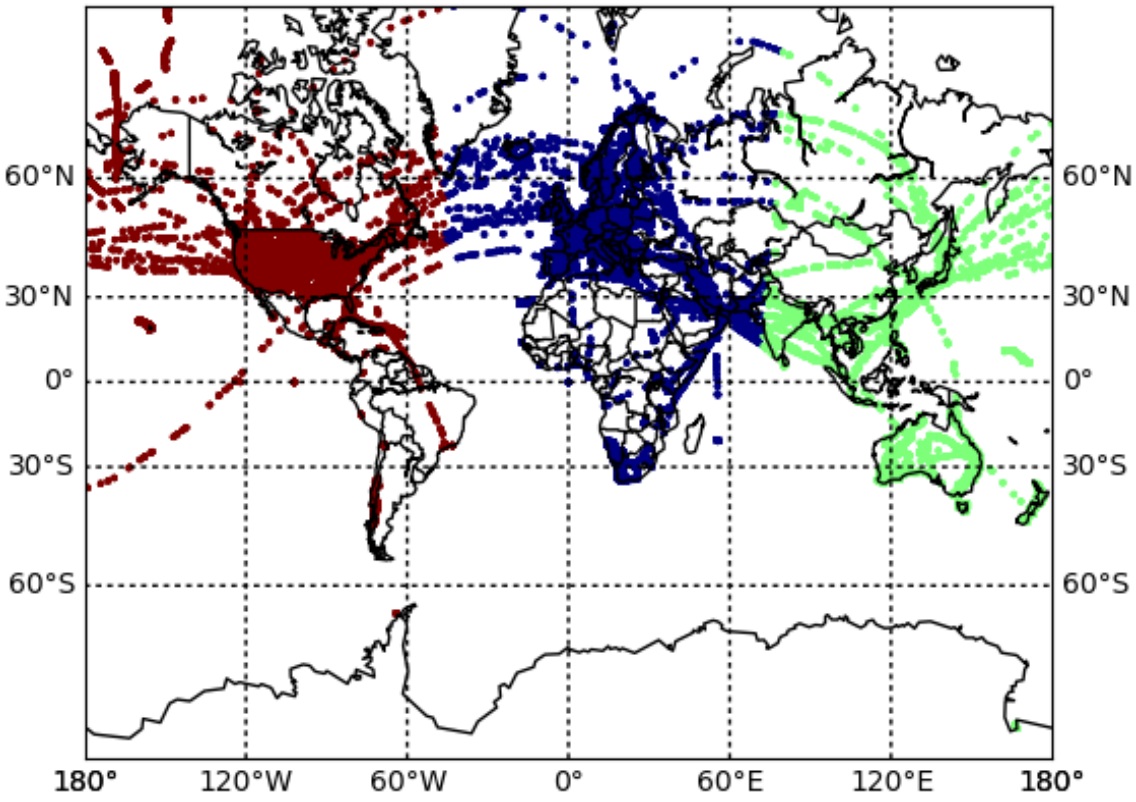


Figure 5.3: The clusters that have formed after performing k-means clustering on the BDAD, the different colours representing the three clusters that have formed, for radiations readings across the earth.

The results therefore show the number of clusters is dependent on the desired insights to be gained. Examples and modifications to further improve the analysis, is to remove the data points that were taken by air or sea, shown in Figure 5.4, by the arching data points. This could prevent these data points which are not covering a country from forming separate clusters, which do not represent a country (an assumption that could be made). Another approach could be to make the number of clusters equal to the number of nuclear reactors found world-wide, to determine the radius of radiation given off by each reactor (thereby providing recommendations of a 'safe' distance to live from the reactors). These examples therefore show the number of clusters could be much more than the suggested best of three clusters. The results also show the ability of the BDAD to conduct k-means clustering on this large dataset.

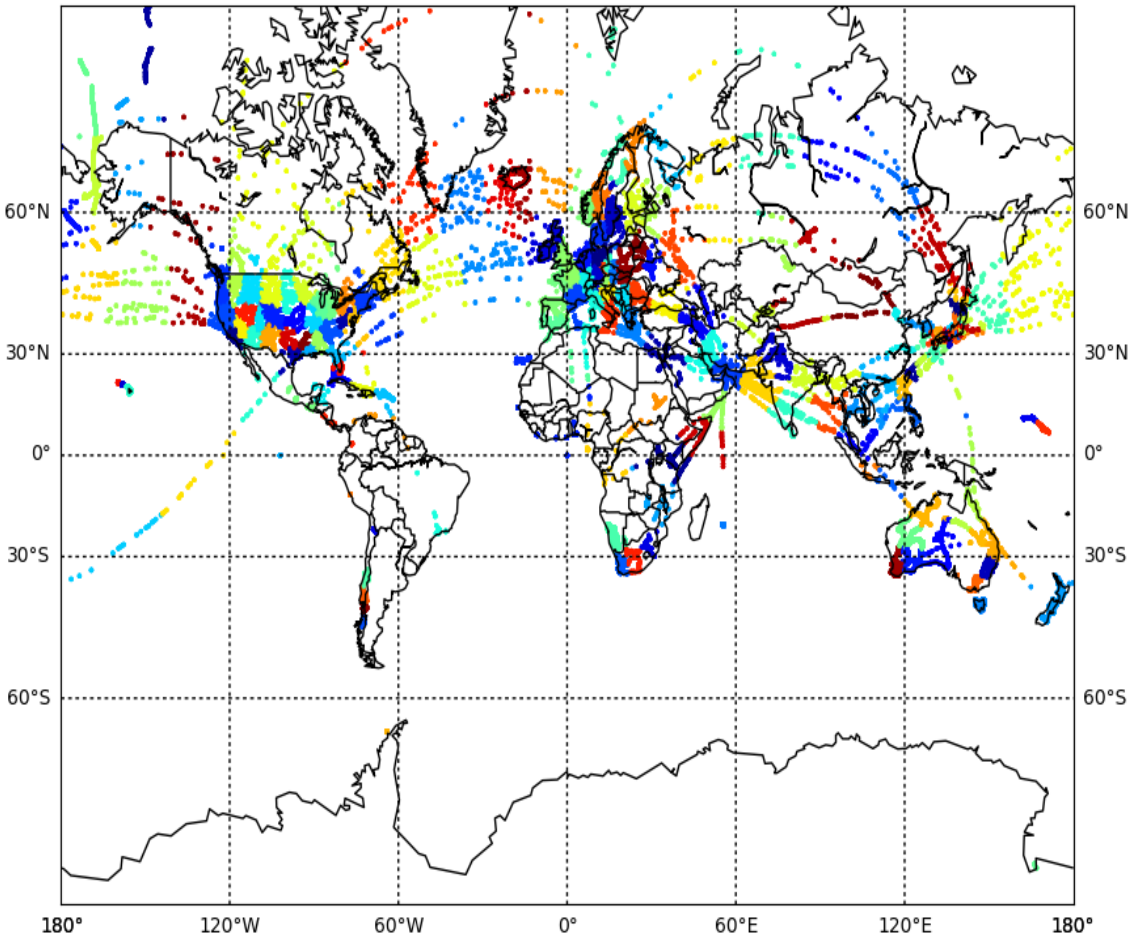


Figure 5.4: The clusters that have formed after performing k-means clustering on the BDAD, the different colours representing the 195 clusters that have formed.

An equivalent test could not be run on the standard system to evaluate the BDAD against, as the standard system was unable to process the large dataset, due to memory errors. This therefore illustrates the benefits of using the BDAD in order to analyse larger datasets in a timely manner.

5.8 Conclusion of the experimental results

This chapter described the experiments conducted with the BDAD, to demonstrate the system's capabilities. This BDAD was developed in order to demonstrate the capabilities of BDA compared to systems which use non-Big Data technologies for data analysis, thereby providing a demonstration of these Big Data technologies. This chapter considered two analytic systems, namely the BDAD and a standard, non-Big Data Analytics system. The two systems were compared by considering the time and accuracy of analysis of different machine learning techniques. The ML algorithms (techniques) were firstly considered, this included different regression algorithms and classification algorithms in order to compare the ability of the two systems to perform *supervised* ML, followed by clustering which is *unsupervised* ML. After this, large datasets that were freely available were identified, each having different data-types suited to the different ML algorithms. The datasets and sizes were chosen to find the analytic limits of both the BDAD and standard system. For regression, the datasets had to have labelled, continuous and numerical features, whereas classification algorithms required labelled, numerical and categorical values in the datasets. Clustering required the data to be un-labelled.

The next part of this chapter considered the KDD analytics process that was discussed in Chapter 2, to guide the analysis process. The first five steps were followed for all ML algorithms, after which steps six to nine were executed depending on the ML algorithm and dataset.

Table 5.12 provides an overview of the results from each of the experiments for each of the ML algorithms, on the BDAD and standard systems respectively.

Table 5.12: A summary of the experimental results from the analysis, using different ML algorithms, to demonstrate the BDAD against the SS.

ML Algorithm	BDAD	Standard System
Logistic Regression	High accuracy in predictions, with 0.3818% lower predictions on average compared to the SS. Able to perform the analysis on all sample sets. Increase in accuracy as sample set sizes were increased. On average 120.1697 seconds per analysis.	Higher accuracies for smaller sample sets and completed an analysis in an average time of 4.6425 seconds. Unable to perform the analysis for sample sets larger than 4 000 000 rows.
Linear Regression	Results indicated a 8.6120% lower prediction accuracy to the SS. The BDAD conducted the analysis for all sample sets, with low $\mu_{\%,\text{BD}} = 0.0187\%$ and taking 4.1526 seconds on average respectively.	The SS had higher predictive accuracies, with $\mu_{\%,\text{SS}} = 92.3230\%$ on average. The SS performed the analyses' on average in 0.2858 seconds, but was unable to perform analyses on sample sets larger than 20 000 000 rows.
Decision Trees	High predictive accuracies, on average 5.3888% better than the SS. The time was 100.2202 seconds on average per analysis, able to analyse all sample sets.	On average 4.0924 seconds per analysis, with higher standard deviations than the BDAD. Global average in predictive accuracy of 68.0359%, and unable to analyse sample sets larger than 20 000 000 rows.
Naive Bayes	Lower predicted accuracies as sample set size increased. Possible cause outlined to be over-or-underfitting. Results remained closer around the mean, with low standard deviations. Able to perform the analysis on all sample sets.	Higher predictive accuracies, on average 21.7936%. Performed the analysis in less time, unable to analyse sample sets larger than 20 000 000 rows.

Continued on next page

Table 5.12 – continued from previous page

ML Algorithm	BDAD	Standard System
Linear SVM	Predictive accuracies of on average 92.6545%, taking longer per analysis, on average 130.7193 seconds. Able to perform the analysis on all sample sets, with low standard deviations.	Performed the analysis faster, for the respective sample sets, with an average time of 2.4798 seconds. More accurate with smaller sample sets, 6.6085% higher on average per analysis. Unable to analyse sample sets larger than 1 000 000 rows.
Multilayer Peceptron Classifier	High predictive accuracies, 73.7873% on average. Time per analysis significantly longer, with 1258.6006 seconds on average, lower standard deviations than the SS.	Predictive accuracies on average 6.2211% lower, faster analysis time on average 30.1457 seconds per analysis. Unable to perform the analysis on sample sets larger than 5 000 000 rows.
k-means	Able to perform the k-means analysis, with the results indicating three clusters to be best number, but as outlined, given the context, clusters of more than three were better suited to the dataset, illustrating as an example, 195 clusters to represent the number of clusters per country (cluster according to country, number of nuclear power stations).	Unable to perform the k-means analysis on the radiation dataset.

These experiments show the use case of BDA and the abilities of the BDAD in performing BDA and ML on large datasets, compared to a standard system using non-Big Data technologies. This also validates the BDAD, and shows the successful implementation of these Big Data technologies to provide a means of

analysing larger datasets. The successful conclusion of these experiments therefore also fulfils the Objectives IV and V as set out in Chapter 1, as the tool was validated and its capabilities were demonstrated to the industrial engineering community using freely available data, with possible applications.

The final chapter considers and summarises the overall result of the project, the future research possible, including successful aspects and shortcomings within the research project.

Chapter 6

Conclusion on the Big Data Analytics Demonstrator

In the previous chapter, the BDAD was validated using different datasets and ML algorithms. This was to demonstrate the ability of BDA compared to non-Big Data Analytics systems. The results were performed using the KDD analytics technique, employing datasets that were appropriate for the specific ML algorithm.

This chapter provides a summary of the research that had been conducted in the field of Big Data, and the development of the BDAD to demonstrate Big Data Analytics and its benefits compared to an analytics system which does not make use of Big Data technologies. Following from this, future research is proposed, and finally a short discussion on various strengths and shortcomings of this research project are presented.

6.1 Summary of the project research and results

This research project was proposed in order to gain a better understanding of what Big Data is and comprises of, through the development of a demonstrator for the industrial engineering community. Large volumes of data are being generated and shared by industries and consumers, and the adoption of Big Data technologies is now more prevalent than before. The need to collect and then anal-

yse this data for the enrichment of users' experience or gaining better insights by improved decision-making for businesses, helps drive this Big Data expansion.

The researcher therefore started the research by conducting a literature review on Big Data and how it is defined; Big Data architectures in literature and industry (case studies), Big Data Analytics (BDA) and defining BDA after holding a focus group with other members of the USMA research group in the industrial engineering department of Stellenbosch University. It was studied how BDA are used to analyse the growing amount of data. Further literature research was then conducted on various open-source systems available to store and process Big Data (the HDFS and Spark systems), the background on the semantic web, RDFs and finally, the benefits, drawbacks and trends in Big Data.

After this, a proposed architecture was developed using the Object Process Methodology (OPM). The proposed architecture was validated using the literature previously researched on Big Data architectures. This architecture guided the development of the demonstrator. In the next phase of the project, the BDAD was built, outlining the process in order to configure three computing nodes together and using HDFS to store the data and Spark to analyse the data. Together with this, considerations that have to be taken into account when creating such a system were also provided. The next step, after the BDAD (consisting of the three computing nodes) had been built, was to test and validate the BDAD. By doing so, the BDAD and therefore Big Data could be demonstrated to the industrial engineering community. To perform the validation, the BDAD was tested and compared against a standard system which does not use Big Data technologies in order to perform the analysis.

This *standard* (non-Big Data) system used the master node, which was disconnected from the cluster and using standard (non-Big Data) Python machine learning libraries to analyse data. The validation was done by recording and then comparing the time and accuracy performance for each ML algorithm (excluding k-means) on the different sample sets of the datasets stored in HDFS. The ML algorithms were chosen from a selection of *supervised* and *unsupervised* ML algorithms, and the different datasets were chosen to be used by these, to evaluate the BDAD's ability to perform regression, classification and clustering analyses. The analytic process chosen was the KDD process due to its generic nature. The

datasets chosen to be used in the project allowed for the different ML algorithms to be tested (each requiring data of a specific data type, for example *categorical*), whilst being large enough to be considered 'Big Data'.

The results from the experiments indicated for logistic regression that the BDAD was able to provide accurate results, but they were on average lower than the SS. The SS when performing the analysis was, however, unable to perform the analysis with sample sets larger than 4 000 000 rows. Linear regression showed that the SS provided higher predictive accuracies, but was also unable to perform the analysis on sample sets larger than 20 000 000 rows compared with the BDAD. Considering the results from the classification analysis, performing the DT analysis, the BDAD on average performed better than the SS, while with naive bayes, the BDAD suffered from either over-or-underfitting, resulting in lower predictive accuracies compared to the SS. Considering the linear SVM analysis, the BDAD had lower predictive accuracies for smaller sample sets, but the SS was unable to perform the analysis with sample sets larger than 1 000 000 rows.

The MPC analysis results indicated higher predictive accuracies on average for the BDAD, but it took significantly longer per analysis on average compared to the SS. The SS was also unable to perform this analysis for sample sets larger than 5 000 000 rows. The BDAD was also able to perform a cluster analysis, providing a means to evaluate the best number of clusters, but the number of clusters could be adjusted given the context of the data (adding clusters to the data to better interpret the dataset). The SS was unable to perform the cluster analysis. From the overall results, the BDAD was able to perform the analysis on all sample sets, with some ML algorithms performing better on the BDAD (as sample set size increased) and demonstrated the systems and capabilities of Big Data. This is in contrast with the SS, which with most ML algorithms was unable to perform the analysis on the largest sample set or had higher predictive accuracies for smaller sample sets (number of rows).

The results illustrated the usefulness and capabilities of the BDAD when compared to a standard analytics system (performing the cluster analysis compared with the SS being unable to do so), when working with large datasets, both for *supervised* and *unsupervised* ML, and BDA. The experiments validated the BDAD as a tool, and demonstrated Big Data and the capabilities of BDA to

the industrial engineering community, validating the project, and fulfilling the Objectives I to V outlined in Chapter 1.

6.2 Future research in Big Data

The development of the BDAD, was limited to analysing datasets in a static, *batch processing* manner against non-Big Data Analytic solutions.

Given that, potential future research can be to expand the BDAD to ingest and analyse data from SQL databases and transfer this data to a distributed non-relational (opposite of SQL) database such as MongoDB. This would allow for the demonstration of using typical industry non-relational database storage methods together with BDA.

Further research could be conducted on demonstrating the use of BDA on other different data types (semi-structured and unstructured), as well as demonstrating the benefits of BDA when conducting a near real time analysis, using the appropriate technologies such as *Cassandra* (designed for fast access, suitable when working with near real time data) instead of HDFS, which is designed for batch analysis. Finally, the Demonstrator could be tested using other ML algorithms found on Spark, not demonstrated in this project.

The recommendations provided for future work to be pursued in Big Data and of the BDAD, meet the final Objective VI. This project has therefore successfully met the objectives and goals, as well as conforming to the defined scope as set out.

6.3 Self-assessment of the research conducted

The research on Big Data and Big Data Analytics in this study provides a concise collection of current knowledge. The project was able to successfully provide a working demonstration tool which uses open-source Big Data software and commodity hardware to perform Big Data Analtyics on *structured* datasets in a *batch-processing manner*, using various ML techniques available. The tool was aimed at giving the industrial engineering domain an idea of the basic development, capabilities, and some practical issues when working with BDA. This

research project also provides a detailed outline of the steps required in order to develop and replicate the BDA (establishing connections and installing the software on the three nodes), which was a non-trivial and time consuming process for the researcher to master, and implement. This is because it required more time than initially thought by the researcher, due to the complex nature of network connections (learning the working thereof) and finding the system's suitable program (Spark and Hadoop) installation configuration, from many possible configurations available (because of the open-source nature). Creating the BDAD was therefore an important milestone that was achieved, requiring the learning of skills outside the industrial engineering environment.

The desire during the initial stages of the project was to develop a BDA system which would be able to ingest multiple datasets of any type (from *structured* to *unstructured*), and perform the analysis (using the appropriate learning technique), without a large amount of user input required, via dashboards *etc.* The interpretation of the results would then be done by using the dashboards and selected visualisations.

Following the development of the BDAD, and working with the open-source software, it was discovered this would be impractical and difficult, as depending on the data types and analysis desired, each analysis had to be individually coded, in order to learn and extract the results from the data. Then only could the individual visualisations of the results be developed and interpreted. Due to this, the researcher decided it was better to develop the BDAD to store all the datasets and analyse (using the appropriate ML technique) each individually with individual programs, after which visualisations were made to interpret the results from each analysis.

The researcher learnt from conducting this study that Big Data and Big Data Analytics are powerful tools, and that BDA systems can encapsulate a wide array of configurations (programs, languages, OSs *etc.*) depending on the application. Finally, Big Data Analytics should be applied to a specific application (data type and/or query), instead of developing a system to analyse for all possibilities, queries and data types.

The satisfaction of the research and the experience gained in this project convinced the researcher to pursue a career in Big Data Analytics.

References

H. Abdi. The method of least squares. *Studia Geophysica et Geodaetica*, 14(2): 107–109, 2007. ISSN 00393169. 54

C. C. Aggarwal and C. K. Reddy. *Data Custering: Algorithms and Applications*. 2014. ISBN 1466558210 9781466558212. 82, 83, 84, 87, 88, 89, 90

C. C. Aggarwal and J. Wang. Data Streams: Models and Algorithms. *Data Streams*, 31:9–38, 2007. DOI: http://dx.doi.org/10.1007/978-0-387-47534-9. 17

D. Agrawal, P. Bernstein, B. Elisa, D. Susan, D. Umeshwar, F. Michael, G. Johannes, H. Laura, H. Alon, H. Jiawei, H. V. Jagadish, L. Alexandros, M. Sam, J. M. Papakonstantinou, Yannis Patel, R. Raghu, K. Ross, S. Cyrus, S. Dan, V. Shiv, and J. Widom. Challenges and Opportunities with Big Data. *Proceedings of the VLDB Endowment*, pages 1–15, 2012. ISSN 2150-8097. DOI: http://dx.doi.org/10.14778/2367502.2367572. 15, 21

D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist*. Morgan Kaufmann, 225 Wyman Street, Waltham, MA 02451, USA, 2nd edition, 2011. ISBN 9780123859655. 115, 116, 117

E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, London, England, 2nd edition, 2010. ISBN 9780262012430. 46, 81, 82, 98, 170

P. Andritsos. Data clustering techniques. Available at: ftp://ftp.cs.toronto.edu/cs/ftp/pub/reports/csri/443/depth.pdf, 2002. Online, Accessed: 05 February 2017. 85, 86, 87, 91, 92

V. Ankam and M. Guller. *Big Data Analytics with Spark*. Packt Publishing, New York, USA, 1st edition, 2015. ISBN 978-1-4842-0965-3. 111, 113

A. S. Association. Miscellaneous Datasets. Available at: http://www.stat.ufl.edu/{~}winner/data/airq402.dathttp://www.stat.ufl.edu/{~}winner/data/airq402.txthttp://www.stat.ufl.edu/{~}winner/datasets.html, 2016a. Online, Accessed: 06 April 2017. 51

A. S. Association. Miscellaneous Datasets 2. Available at: http://www.stat.ufl.edu/{~}winner/datasets.htmlhttp://www.stat.ufl.edu/{~}winner/data/ship{_}speed{_}fuel.txt, 2016b. Online, Accessed: 06 April 2017. 55

V. Astakhov and M. Chayel. Lambda Architecture for Batch and Real- Time Processing on AWS with Spark Streaming and Spark SQL. Available at: https://d1.awsstatic.com/whitepapers/lambda-architecure-on-for-batch-aws.pdf, 2015. Online, Accessed: 10 May 2017. 31, 136, 143

T. O. Ayodele. Machine Learning Overview. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.800.5174&rep=rep1&type=pdf, 2010. Online, Accessed: 30 March 2017. 44, 45

A. Azevedo and M. F. Santos. KDD, SEMMA and CRISP-DM: A Parallel Overview. Available at: http://recipp.ipp.pt/bitstream/10400.22/136/3/KDD-CRISP-SEMMA.pdf, 2008. Online, Accessed: 25 March 2017. 37, 39, 40

B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*, page 1, 2002. ISSN 1581135076. DOI: http://portal.acm.org/citation.cfm?doid=543613.543615. 17

K. Bakshi. Considerations for big data: Architecture and approach. *IEEE Aerospace Conference Proceedings*, pages 1–7, 2012. DOI: https://doi.org/10.1109/AERO.2012.6187357. 104, 105

W. Balke. Introduction to Information Extraction: Basic Notions and Current Trends. *Datenbank-Spektrum*, 12(2):81–88, 2012. ISSN 1618-2162. DOI: https://doi.org/10.1007/s13222-012-0090-x. 18

A. G. Barto and S. Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. Available at: http://www-anw.cs.umass.edu/pubs/2003/barto_m_DEDS03.pdf, 2003. Online, Accessed: 29 March 2017. 46, 98

I. A. Basheer and M. Hajmeer. Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3–31, 2000. ISSN 01677012. DOI: https://doi.org/10.1016/S0167-7012(00)00201-3. 70, 71, 81

D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*. Wiley Series in Probability and Statistics. Wiley, 1988. ISBN 9780470139004. 49

V. Beal. Webopedia. Available at: www.webopedia.com/TERM/L/library.html, 2018. Online, Accessed: 18 March 2018. 161

B. Bearnes. Package Management Basics: apt, yum, dnf, pkg. Available at: https://www.digitalocean.com/community/tutorials/package-management-basics-apt-yum-dnf-pkg, 2018. Online, Accessed: 25 February 2018. 226

A. Bekker. Spark vs. Hadoop MapReduce: Which big data framework to choose. Available at: https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce, 2017. Online, Accessed: 01 March 2018. 111, 112, 156

Berkhin. Survey Of Clustering Data Mining Techniques. *Accrue Software, San Jose, CA*, pages 1–56, 2002. DOI: https://doi.org/10.1007/3-540-28349-8_2. 85, 89, 91

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. Available at: http://www.jstor.org/stable/26059207, 2001. Online, Accessed: 03 April 2017. 115, 116

A. Berson, S. Smith, and K. Thearling. An Overview of Data Mining Techniques. Available at: http://weber.itn.liu.se/~jimjo94/courses/TNM048/documents/DM-Techniques.pdf, 2005. Online, Accessed: 21 March 2017. 45

N. Bhatia and V. Ashev. Survey of Nearest Neighbor Techniques. *IJCSIS) International Journal of Computer Science and Information Security*, 8(2): 302–305, 2010. ISSN 1947-5500. Available at: https://arxiv.org/abs/1007.0085. 78, 80

L. Bijuraj. Clustering and its Applications. 2013. ISBN 978-93-82338-79-6. Available at: http://www.conference.bonfring.org/papers/met_ncnhit2013/ncnhit40.pdf. 183

C. M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer, 2013. ISBN 978-0387-31073-2. 48

J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 41(1): 1–41, 2008. DOI: https://doi.org/10.1145/1456650.1456651. 19

J. Z. Bloom. Tourist market segmentation with linear and non-linear techniques. *Tourism Management*, 25(6):723–733, 2004. ISSN 02615177. DOI: https://doi.org/10.1016/j.tourman.2003.07.004. 63

R. J. Bolton and H. D. J. Unsupervised Profilinig Methods for Fraud Detection. Available at: https://pdfs.semanticscholar.org/5b64/0c367ae9cc4bd072006b05a3ed7c2d5f496d.pdf, 2018. Online Accessed: 04 June 2018. 183

D. Bonino and L. De Russis. Mastering real-time big data with stream processing chains. *XRDS*, 19(1):83–86, 2012. ISSN 1528-4972. DOI: https://doi.org/10.1145/2331042.2331050. 17

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-555-X. 93

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. DOI: https://doi.org/10.1023/A:1010933404324. 62

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN 9780412048418. 62

A. Bridgwater. Why Linux is the powerhouse for big data. Available at: http://www.computerweekly.com/blog/Open-Source-Insider/Why-Linux-is-the-powerhouse-for-big-data, 2013. Online, Accessed: 05 February 2018. 153

C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery 2*, 43:1–43, 1998. DOI: https://doi.org/10.1023/A:1009715923555. 72, 73, 81

R. D. Butler and J. Bekker. Development of a Big Data Analysis Demonstrator. In J. Bekker, editor, *SAIIE 28th Annual Conference*, pages 1–13, Stellenbosch, South Africa, 2017. 13, 37, 43, 119, 124, 136, 137

P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi. *Discovering Data Mining: From Concept To Implementation*. Prentice Hall, New Jersey, NY, USA, 1st edition, 1998. ISBN 0-13-743980-6. 40, 41, 42

O. Cakir and M. E. Aras. A Recommendation Engine by Using Association Rules. *Procedia - Social and Behavioral Sciences*, 62:452–456, 2012. ISSN 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2012.09.074. 64

D. Campbell and S. Campbell. Introduction to Regression and Data Analysis. Available at: http://statlab.stat.yale.edu/workshops/IntroRegression/StatLab-IntroRegressionFa08.pdf, 2008. Online, Accessed: 01 June 2018. 170, 175

K. S. Candan, H. Liu, and R. Suvarna. Resource Description Framework: Metadata and Its Applications. 3(1), 2001. DOI: https://doi.org/10.1145/507533.507536. 117

C.-C. H. Chan. Online auction customer segmentation using a neural network model. *International Journal of Applied Science and Engineering*, 3(2):101–109, 2005. ISSN 1727-2394. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.540.5344. 63

S. Chatterjee and A. S. Hadi. *Regression Analysis by Example*. Wiley Series in Probability and Statistics. Wiley, 4th edition, 2006. ISBN 9781118456248. Available at: https://books.google.co.za/books?id=86MCAZaY1noC. 49

S. Chiu and D. Tavella. Chapter 7 - Introduction to Data Mining. In S. Chiu and D. Tavella, editors, *Data Mining and Market Intelligence for Optimal Marketing Returns*, pages 137–192. Butterworth-Heinemann, Boston, 2008. ISBN 978-0-7506-8234-3. DOI: http://dx.doi.org/10.1016/B978-0-7506-8234-3.00007-1. 84, 85

Cloudera. Spark Guide. Available at: https://www.cloudera.com/documentation/enterprise/5-6-x/PDF/cloudera-spark.pdf, 2017. Online, Accessed: 01 December 2017. 110, 112, 113, 114

T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. MapReduce Online. Available at: http://static.usenix.org/events/nsdi10/tech/full_papers/condie.pdf, 2010. Online, Accessed: 05 February 2018. 108

K. Coussement and D. den Poel. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert Systems with Applications*, 34(1):313–327, 2008. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2006.09.038. 63

Y. Darji. Easily Setup Multi-Node Hadoop Cluster in YARN Mode on CentOS/Ubuntu. Available at: https://medium.com/@yogeshdarji99/installing-multinode-hadoop-cluster-in-yarn-mode-86741c3df45b, 2017. Online, Accessed: 27 February 2018. 229, 231

T. K. Das and P. M. Kumar. Big Data Analytics : A Framework for Unstructured Data Analysis. *International Journal of Engineering and Technology*, 5(1):153–156, 2013. ISSN 0975-4024. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.411.6697&rep=rep1&type=pdf. 104

DataDotz. Spark Master/Slave installation in Multi Node. Available at: http://chennaihug.org/knowledgebase/spark-master-and-slaves-multi-node-installation/, 2018. Online, Accessed: 02 March 2018. 247

Dell. Dell Optiplex 9020 Specifications. Available at: https://www.dell.com/learn/us/en/04/shared-content~data-sheets~en/documents~dell-optiplex-9020-spec-sheet_final_v2_g13001038.pdf, 2018a. Online, Accessed: 17 April 2018. 148

Dell. Dell Precision T3500 Specifications. Available at: http://www.dell.com/downloads/global/products/precn/en/dell{_}precision{_}t3500{_}specsheet.pdf, 2018b. Online, Accessed: 17 April 2018. 148

J. Demšar and B. Zupan. Orange: Data mining fruitful and fun: A historical perspective. *Informatica*, 37(1), 2013. Available at: http://www.informatica.si/ojs-2.4.3/index.php/informatica/article/viewFile/434/438. 84

DeZyre. Step-by-Step Apache Spark Installation Tutorial. Available at: https://www.dezyre.com/apache-spark-tutorial/apache-spark-installation-tutorial, 2018. Online, Accessed: 03 March 2018. 247

V. Dhote, S. Mishra, G. Parajapati, and J. Shukla. Challenges in Big Data Application: A Review. *International journal of computer applications*, 121 (19):42–46, 2015. DOI: https://doi.org/10.5120/21651-4962. 105

T. G. Dietterich. Machine-Learning Research: Four Current Directions. *AI Magiazine*, 18(4):97–136, 1997. DOI: https://doi.org/10.1609/aimag.v18i4.1324. 98, 99

J. Dittrich and J.-a. Quian. Efficient Big Data Processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment*, 5(12):2014–2015, 2012. ISSN 21508097 (ISSN). DOI: https://doi.org/10.14778/2367502.2367562. 105, 108

D. Dov. *Object-process methodology: A holistic systems paradigm*. Springer Science + Business Media LLC. DOI: https://doi.org/10.1007/978-3-642-56209-9, Year = 2011. 136

DWBI. How to Setup Hadoop Multi Node Cluster - Step By Step. Available at: https://dwbi.org/etl/bigdata/183-setup-hadoop-cluster, 2016. Online, Accessed: 27 February 2018. 229, 231

K. Dwivedi and S. K. Dubey. Analytical review on Hadoop Distributed file system. *Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit*, pages 174–181, 2014. DOI: https://doi.org/10.1109/CONFLUENCE.2014.6949336. 105, 109

M. D. Ekstrand. Collaborative Filtering Recommender Systems. *Foundations and Trends in HumanComputer Interaction*, 4(2):81–173, 2011. ISSN 1551-3955. DOI: https://doi.org/10.1561/1100000009. 92, 93, 94, 95, 96, 97

J. Ellingwood. Hadoop, Storm, Samza, Spark, and Flink: Big Data Frameworks Compared. Available at: https://www.digitalocean.com/community/tutorials/hadoop-storm-samza-spark-and-flink-big-data-frameworks-compared, 2016. Online, Accessed: 17 March 2018. 156, 157

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. *A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. KDD'96. AAAI Press, 1996. Available at: http://dl.acm.org/citation.cfm?id=3001460.3001507. 85, 90, 98

W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *Proceedings of the VLDB Endowment*, 2(1):407–418, 2009. ISSN 2150-8097. DOI: https://doi.org/10.14778/1687627.1687674. 19

U. M. Feyyad. Data mining and knowledge discovery: making sense out of data. *Ieee Expert Intelligent Systems And Their Applications*, 11(5):20–25, 1996. ISSN 08859000. DOI: https://doi.org/10.1109/64.539013. 37, 38

Fibrevillage. Hadoop 2.7 cluster installation and configuration on RHEL7/CentOS7. Available at: http://fibrevillage.com/storage/617-hadoop-2-7-cluster-installation-and-configuration-on-rhel7-centos7, 2016. Online, Accessed: 28 February 2018. 229, 231

T. Fisher. What is Bandwidth. Available at: https://www.lifewire.com/what-is-bandwidth-2625809, 2018. Online, Accessed: 19 February 2018. 150

T. A. S. Foundation. Apache Hadoop. Available at: http://hadoop.apache.org/, 2014. Online, Accessed: 06 June 2017. 103, 105, 106, 107

A. R. Gallant. Nonlinear Regression. *The American Statistician*, 29(2):73–81, 1975. ISSN 00031305. DOI: https://doi.org/10.2307/2683268. 49

M. Galster and P. Avgeriou. Empirically-grounded Reference Architectures : A Proposal. pages 153–157, 2011. DOI: https://doi.org/10.1145/2000259.2000285. 13, 133

M. Gera and S. Goel. Data Mining -Techniques, Methods and Algorithms: A Review on Tools and their Validity. *International Journal of Computer Applications*, 113(18):22–29, 2015. ISSN 09758887. DOI: https://doi.org/10.5120/19926-2042. 42, 44, 48, 49

M. R. Ghazi and D. Gangodkar. Hadoop, mapreduce and HDFS: A developers perspective. *Procedia Computer Science*, 48(C):45–50, 2015. ISSN 18770509. DOI: https://doi.org/10.1016/j.procs.2015.04.108. 107, 108, 110

D. Gil and J. L. Girela. UCI Machine Learning Repository. Available at: http://archive.ics.uci.edu/ml/datasets/Fertility, 2013. Online, Accessed: 26 January 2018. 58

P. Gleeson. Which Languages Should You Learn For Data Science? Available at: https://medium.freecodecamp.org/which-languages-should-you-learn-for-data-science-e806ba55a81f, 2017. Online, Accessed: 05 March 2018. 158

V. Granville. Data Science Cheat Sheet. Available at: https://www.datasciencecentral.com/profiles/blogs/data-science-cheat-sheet, 2014. Online, Accessed: 18 February 2018. 153

M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems.*, 17(2):107–145, 2001. ISSN 0925-9902. DOI: https://doi.org/10.1023/A:1012801612483,. 85

P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1-3):191–215, 1997. ISSN 0025-5610. DOI: https://doi.org/10.1007/BF02614317. 183

J. J. Hanson. An introduction to the Hadoop Distributed File System. Available at: https://www.ibm.com/developerworks/library/wa-introhdfs/index.html, 2011. Online, Accessed: 16 August 2017. 105

F. E. Harrell. *Regression Modeling Strategies*, volume 64. Springer Series in Statistics, 2nd edition, 2015. ISBN 978-1-4419-2918-1. DOI: https://doi.org/10.1007/978-1-4757-3462-1. 51, 52, 57, 60

I. A. T. Hashem, N. B. Anuar, A. Gani, I. Yaqoob, F. Xia, and S. U. Khan. MapReduce: Review and open challenges. *Scientometrics*, 109(1):389–422, 2016. ISSN 15882861. DOI: https://doi.org/10.1007/s11192-016-1945-y. 107, 108, 109

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. *Elements*, 1:337–387, 2009. ISSN 03436993. DOI: https://doi.org/10.1007/b94608. 63

J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd annual*

international ACM SIGIR conference on Research and development in information retrieval - SIGIR '99, pages 230–237, 1999. ISSN 09540121. DOI: https://doi.org/10.1145/312624.312682. 93

D. Hosmer and S. Lemeshow. *Applied Logistic Regression*. Wiley-Interscience, Hoboken, New Jersey, 3rd edition, 2013. ISBN 9780470582473. DOI: https://doi.org/10.1002/9781118548387. 49, 51, 57, 58, 60

B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali. A comparative study of decision tree ID3 and C4.5. *International Journal of Advanced Computer Science and Applications*, (2):13–19, 2014. ISSN 2158107X. DOI: https://doi.org/10.14569/SpecialIssue.2014.040203. 62

J.-J. Huang, G.-H. Tzeng, and C.-S. Ong. Marketing segmentation using support vector clustering. *Expert systems with applications*, 32(2):313–317, 2007. DOI: https://doi.org/10.1016/j.eswa.2005.11.028. 63

IntelliPaat. Hadoop vs Spark Choosing the Right Big Data Software. Available at: https://intellipaat.com/blog/hadoop-vs-spark-choosing-the-right-big-data-software/, 2016. Online, Accessed: 07 March 2018. 156

H. Ishibuchi and T. Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *IEEE transactions on fuzzy systems*, 13(4):428–435, 2005. DOI: https://doi.org/10.1109/TFUZZ.2004.841738. 64

A. J. Izenman. *Modern Multivariate Statistical Techniques: regression, classification, and Manifold Learning*. 2008. ISBN 9780387781884. DOI: https://doi.org/10.1007/978-0-387-78189-1. 63, 84, 85

S. G. Jacob and R. G. Ramani. Data mining in clinical data sets: a review. *Foundation of Computer Science FCS, New York, USA*, 4(6), 2012. ISSN 2249-0868. DOI: https://doi.org/10.5120/ijais12-450774. 84

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. ISSN 03600300. DOI: https://doi.org/10.1145/331499.331504. 45, 86, 87

K. Jain. 11 things you should know as a Data Scientist. Available at: https://www.analyticsvidhya.com/blog/2015/08/ready-data-science-resources-common-questions-answered/, 2015. Online, Accessed: 05 February 2018. 153

S. M. H. Jansen. Customer Segmentation and Customer Profiling for a Mobile Telecommunications Company Based on Usage Behavior: A Vodafone Case Study. Available at: https://www.semanticscholar.org/paper/Customer-Segmentation-and-Customer-Profiling-for-a/7a3a688783e0424bd89f7413138bbfc24deeef8f?tab=citations, 2007. Online, Accessed: 11 May 2017. 63, 84

M. Jeevan. How I chose the right programming language for Data Science. Available at: http://bigdata-madesimple.com/how-i-chose-the-right-programming-language-for-data-science/, 2015. Online, Accessed: 23 April 2018. 156, 158

F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Series in Statistics, Aalborg, Denmark, 2nd edition, 2007. ISBN 9780387682815. 74, 75, 76, 81

L. Jiang, Z. Cai, D. Wang, and S. Jiang. Survey of improving K-nearest-neighbor for classification. *Proceedings - Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*, 1:679–683, 2007. DOI: https://doi.org/10.1109/FSKD.2007.552. 78, 79, 80, 81

H. Jiawei, M. Kamber, J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. 3rd edition, 2012. ISBN 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00001-0. 18, 37, 40, 42, 43, 60, 63, 65, 66, 80, 84, 175

I. T. Jolliffe. Principal component analysis. *Applied Optics*, 44(May):6486, 2005. ISSN 01621459. DOI: https://doi.org/10.1007/978-3-642-04898-2_455. 100

M. T. Jones. Process real-time big data with Twitter Storm- An introduction to streaming big data. Available at: https://www.ibm.com/developerworks/library/os-twitterstorm/index.html, 2012. Online, Accessed: 04 February 2017. 26, 27

K. Kambatla, G. Kollias, V. Kumar, and A. Grama. Trends in big data analytics. *Journal of Parallel and Distributed Computing*, 74(7):2561–2573, 2014. ISSN 07437315. DOI: https://doi.org/10.1016/j.jpdc.2014.01.003. 3, 119, 121

J. K. Kamdar, P. M. S. Patil, and C. B. Khatri. Big Data - An Overview. *International Journal of Engineering Research & Technology (IJERT)*, 3(7): 80–83, 2014. ISSN 2278-0181. 22, 24, 27

A. H. Karp. Using logistic regression to predict customer retention. In *Proceedings of the Eleventh Northeast SAS Users Group Conference.*, 1998. Available at: https://pdfs.semanticscholar.org/402a/4dbcff751467652446b25840f76897f8d4f4.pdf?_ga=2.174552226.1834464766.1535202715-1216715367.1535202715. 49

A. Katal, M. Wazid, and R. H. Goudar. Big Data: Issues, Challenges, Tools and Good Practices. pages 404–409, 2013. DOI: https://doi.org/10.1109/IC3.2013.6612229. 11

S.-Y. Kim, T.-S. Jung, E.-H. Suh, and H.-S. Hwang. Customer segmentation and strategy development based on customer lifetime value: A case study. *Expert systems with applications*, 31(1):101–107, 2006. DOI: https://doi.org/10.1016/j.eswa.2005.09.004. 62

M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja. Lambda architecture for cost-effective batch and speed big data processing. *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pages 2785–2792, 2015. DOI: https://doi.org/10.1109/BigData.2015.7364082. 23, 25

J. Kogan, C. Nicholas, and M. Teboulle. *Grouping Multidimensional Data: Recent advances in clustering*. Springer Berlin Heidelberg, Baltimore, USA, 1st

edition, 2006. ISBN 978-3-540-28348-5. DOI: https://doi.org/10.1007/3-540-28349-8. 46, 81, 85, 88, 89, 91, 92

S. Kotsiantis. *Supervised Machine Learning: A Review of Classification Techniques*. 2007. DOI: https://doi.org/10.1007/s10462-007-9052-3. 45, 61, 62, 63, 64, 72, 73, 74, 78, 81, 175, 177

D. Kriesel. Neural Networks. Available at: http://www.dkriesel.com/en/science/neural_networks, 2005. Online, Accessed: 12 May 2017. 71, 81

R. J. Kuo, Y. L. An, H. S. Wang, and W. J. Chung. Integration of self-organizing feature maps neural network and genetic K-means algorithm for market segmentation. *Expert Systems with Applications*, 30(2):313–324, 2006. ISSN 0957-4174. DOI: http://dx.doi.org/10.1016/j.eswa.2005.07.036. 63, 84

D. S. Lab. Finding the K in K-Means Clustering. Available at: https://datasciencelab.wordpress.com/2013/12/27/finding-the-k-in-k-means-clustering/, 2013. Online, Accessed: 28 June 2018. 184

D. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley, 2nd edition, 2014. ISBN 9780471687535. DOI: https://doi.org/10.1002/9781118874059. 62, 64

R. L. Lawrence and A. Wright. Rule-based classification systems using classification and regression tree (CART) analysis. *Photogrammetric engineering and remote sensing*, 67(10):1137–1142, 2001. Available at: https://pdfs.semanticscholar.org/0a3c/ec5d1c1c1ba8ee2ce44dcae1b2bcc93c5519.pdf. 64

K.-h. Lee, Y.-j. Lee, H. Choi, Y. D. Chung, and B. Moon. Parallel Data Processing with MapReduce: A Survey. *ACM SIGMOD Record*, 40(4):11, 2011. ISSN 01635808. DOI: https://doi.org/10.1145/2094114.2094118. 107, 108, 109

J. Leskovec, R. Anand, and J. Ullman. Recommendation Systems. *Mining of Massive Datasets*, pages 305–339, 2011. DOI: https://doi.org/10.1017/CBO9781139924801.010. 93, 95

J. Lever, M. Krzywinski, and N. Altman. Points of Significance: Principal component analysis. *Nature Methods*, 14(7):641–642, 2017. ISSN 15487105. DOI: https://doi.org/10.1038/nmeth.4346. 100

R. Li. Top 10 Data Mining Algorithms, Explained. Available at: http://www.kdnuggets.com/2015/05/top-10-data-mining-algorithms-explained.html, May 2015. Online, Accessed: 16 May 2017. 63, 64

S. Li, Y. Yang, L. Yang, H. Su, G. Zhang, and J. Wang. Civil Aircraft Big Data Platform. *2017 IEEE 11th International Conference on Semantic Computing*, pages 328–333, 2017. DOI: https://doi.org/10.1109/ICSC.2017.51. 33, 34, 143, 144, 145

Linglom.com. Configure static IP address on CentOS 7. Available at: https://www.linglom.com/networking/configure-static-ip-address-on-centos-7/, 2017. Online, Accessed: 08 March 2018. 227

G. S. Linoff and M. J. A. Berry. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. IT Pro. Wiley, 2011. ISBN 9781118087459. Available at: https://books.google.co.za/books?id=AyQfVTDJypUC. 63

X. Liu, N. Iftikhar, and X. Xie. Survey of real-time processing systems for big data. *Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14*, pages 356–361, 2014. DOI: https://doi.org/10.1145/2628194.2628251. 23, 24

S. Lohr. The Age of Big Data. Available at: https://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html, 2012. ISSN 0362-4331. Online, Accessed: 25 February 2017. 2, 4

T. S. Madhulatha. Comparison between k-means and k-medoids clustering algorithms. *Advances in Computing and Information Technology*, pages 472–481, 2011. DOI: https://doi.org/10.1007/978-3-642-22555-0_48. 84, 85

T. V. Maia. Reinforcement learning, conditioning, and the brain: Successes and Challenges. *Cognitive, Affective, & Behavioural Neuroscience*, 9(4):343–364, 2009. DOI: https://doi.org/10.3758/CABN.9.4.343. 46, 99

M. Maier. *Towards a Big Data Reference Architecture*. M.sc, Eindhoven University of Technology, 2013. 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 34, 35, 125, 131, 132, 133, 135, 137, 143, 144

J. Mao. Artificial Neural Networks: A Tutorial. 1996. DOI: https://doi.org/10.1109/2.485891. 72

I. MapR Technologies. MapR. https://mapr.com/developercentral/lambda-architecture/, 2017. Online, Accessed: 23 June 2017. 23

S. Marchal, X. Jiang, R. State, and T. Engel. A Big Data Architecture for Large Scale Security Monitoring. 2014. DOI: https://doi.org/10.1109/BigData.Congress.2014.18. 28, 143

G. Mariscal, Ó. Marbán, and C. Fernández. A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, 25(02):137–166, 2010. ISSN 0269-8889. DOI: https://doi.org/10.1017/S0269888910000032. 37, 39

B. Marr. Ten top languages for crunching Big Data. Available at: https://www.datasciencecentral.com/profiles/blogs/ten-top-languages-for-crunching-big-data, 2015. Online, Accessed: 05 March 2018. 158

N. Marz and J. Warren. *Big Data*. Manning Publications, New York, USA, 1 edition, 2015. ISBN 9781617290343. 22, 23, 24, 25, 26, 27, 35, 133, 135, 143

M. Mayilvaganan and M. Sabitha. A cloud-based architecture for Big-Data Analytics in Smart Grid : A Proposal. *Computer Science Review*, page 4, 2013. DOI: https://doi.org/10.1109/ICCIC.2013.6724168. 31, 32, 143

A. McAfee and E. Brynjolfsson. Big Data. The management revolution. Available at: https://hbr.org/2012/10/big-data-the-management-revolution, 2012. Online, Accessed: 28 January 2017. 1, 11

S. Menard. *Applied Logistic Regression Analysis*. SAGE Publications, Thousand Oaks, London, 2nd edition, 2002. ISBN 9780761922087. DOI: https://doi.org/10.4135/9781412983433. 51, 57

X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar. MLlib : Machine Learning in Apache Spark. *Journal of Machine Learning Research 17*, 17:1–7, 2016. ISSN 1532-4435. 156

P. Mika. Social Networks and the Semantic Web. In *IEEE Computer Society*, pages 1–7. IEE Computer Society, 2004. DOI: https://doi.org/10.1007/978-0-387-71001-3. 115

K. W. Miller and K. Michael. Big Data: New Opportunities and New Challenges. pages 22–24, 2013. DOI: https://doi.org/10.1109/MC.2013.196. 21

S. Mitra. How to Setup Hadoop Multi Node Cluster - Step By Step. Available at: https://dwbi.org/etl/bigdata/201-install-spark-in-hadoop-cluster, 2016. Online, Accessed: 12 March 2018. 247

I. Mohamad and D. Usman. Standardization and Its Effects on K-Means Clustering Algorithm. *Research Journal of Applied Sciences, Engineering and Technology 6(17): 3299-3303*, pages 1–5, 2013. ISSN 20140-7459. 183

D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression*. John Wiley & Sons Inc., Hoboken, New Jersey, 5th edition, 2012. ISBN 978-0-470-54281-1. 49, 51, 52, 53, 57

K. Murphy. A Brief Introduction to Graphical Models and Bayesian Networks. Available at: https://www.cs.ubc.ca/~murphyk/Bayes/bayes_tutorial.pdf, 1998. Online, Accessed: 29 January 2018. 76, 77

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning. MIT Press, 2012. ISBN N 978-0-262-01802-9. 62

D. Mustatea. How to Choose the Right Programming Language for Your Big Data Initiatives. Available at: https://bigstep.com/blog/choose-right-programming-language-big-data-initiatives, 2016. Online, Accessed: 07 March 2018. 158

A. Nagpal. L1 and L2 Regularisation Methods. Available at: https://towardsdatascience.com/l1-and-l2-regularization-metods-ce25e7fc831c, 2017. Online, Accessed: 22 May 2018. 258

D. Napoleon and S. Pavalakodi. A new method for dimensionality reduction using K-means clustering algorithm for high dimensional data set. *International Journal of Computer Applications*, 13(7):41–46, 2011. DOI: https://doi.org/10.5120/1789-2471. 84

D. Nautiyan. Underfitting and Overfitting in Machine Learning. Available at: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/, 2013. Online, Accessed: 25 June 2018. 178

A.-c. N. Ngomo, S. Auer, J. Lehmann, and A. Zaveri. Introduction to Linked Data and its Lifecycle on the Web. *Informatik, Institut Leipzig, Universität*, pages 1–105, 2013. DOI: https://doi.org/10.1007/978-3-642-39784-4_1. 19

N. F. Noy. Semantic Integration : A Survey Of Ontology-Based Approaches. 33 (4):65–70, 2004. DOI: https://doi.org/10.1145/1041410.1041421. 115

M. Oded and R. Lior. *Data Mining and Knowledge Discovery Handbook*. 2005. ISBN 0-387-24435-2. DOI: https://doi.org/10.1007/b107408. 69, 80

F. Ohlhorst. *Big Data Analytics: Turning Big Data into Big Money*. John Wiley and Sons, Mumbai, India, 1st edition, 2015. ISBN 9781119205005. DOI: http://doi.wiley.com/10.1002/9781119205005. 110

M. Paliwal and U. A. Kumar. A study of academic performance of business school graduates using neural network and statistical techniques. *Expert Systems with Applications*, 36(4):7865–7872, 2009. ISSN 09574174. DOI: https://doi.org/10.1016/j.eswa.2008.11.003. 48

V. Paramasivam, T. S. Yee, S. K. Dhillon, and A. S. Sidhu. A methodological review of Data mining techniques in predictive medicine: An application in hemodynamic prediction for abdominal aortic aneurysm disease. *Biocybernetics and Biomedical Engineering*, 34(3):139–145, 2014. ISSN 02085216. DOI: https://doi.org/10.1016/j.bbe.2014.03.003. 62

A. B. Patel, M. Birla, and U. Nair. Addressing Big Data Problem Using Hadoop and Map Reduce. pages 6–8, 2012. DOI: https://doi.org/10.1109/NUICONE.2012.6493198. 105

J. Pearl. *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-51102-9. DOI: http://dl.acm.org/citation.cfm?id=303568.303676. 75

N. Petroulakis and A. Miaoudakis. An Application of Neural Networks in Market Segmentation. *In Proc of 2nd Pan-Hellenic Conference in New Technologies and Marketing (NTM2007)*, 2011. Available at: http://users.ics.forth.gr/~npetro/files/NTM_2007.pdf. 63

C. L. Philip Chen and C. Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275:314–347, 2014. ISSN 00200255. DOI: https://doi.org/10.1016/j.ins.2014.01.015. 117

T. N. Phyu. *Survey of Classification Techniques in Data Mining*, volume I. 2009. ISBN 9789881701220. 65, 78, 79, 80, 81

L. Pierson and J. Porway. *Data Science For Dummies*. Wiley, 2017. ISBN 978-1-118-84152-5. 84, 85

S. Polamuri. Difference between softmax function and sigmoid function. Available at: http://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/, 2017. Online, Accessed: 24 June 2018. 264

D. Prasanna. Languages, Verbosity, and Java. Available at: http://www.informit.com/articles/article.aspx?p=1824790, 2012. Online, Accessed: 05 March 2018. 156

J. R. Quinlan. Induction of Decision Trees. *Centre for Advanced Computing Sciences*, 1:81–106, 1986. ISSN 0885-6125. DOI: https://doi.org/10.1007/BF00116251. 66, 67, 68, 175

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Elibrary online. Elsevier Science, 2014. DOI: https://doi.org/10.1007/BF00993309. 62

Quora. Which is the best operating system to learn Hadoop or big data? Available at: https://www.quora.com/Which-is-the-best-operating-system-to-learn-Hadoop-or-big-data, 2015. Online, Accessed: 10 November 2017. 153

Quora. Which OS is best for data analysis: Windows, Mac OS, or Linux? Available at: https://www.quora.com/Which-OS-is-best-for-data-analysis-Windows-Mac-OS-or-Linux, 2016. Online, Accessed: 10 November 2017. 153

K. Rahul. How to Set Up Hadoop Multi-Node Cluster on CentOS 7/6. Available at: https://tecadmin.net/set-up-hadoop-multi-node-cluster-on-centos-redhat/, 2013. Online, Accessed: 27 February 2018. 229, 231

K. Rahul. How to Install JAVA 8 on CentOS/RHEL 7/6 and Fedora 27/26/25. Available at: https://tecadmin.net/install-java-8-on-centos-rhel-and-fedora/, 2016. Online, Accessed: 28 February 2018. 231, 248

A. Rajarajeswari and R. M. Ravindran. A Comparative Study Of K-Means K-Medoid And Enhanced K-Medoid Algorithms. *International Journal of Advance Foundation and Research in Computer (IJAFRC)*, 2(8):7–10, 2015. ISSN 2348 4853. 84

C. Rao. *Linear statistical inference and its applications*, volume 6. Wiley-Interscience, Pennsylvania, 2nd edition, 1966. ISBN 0-471-70823-2. DOI: https://doi.org/10.1016/0041-5553(66)90024-3. 58

S. Raschka. *Python Machine Learning*. Number 1. 2014. ISBN 978-1787125933. 178

M. D. Rechenthin. *Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction*. Doctor of philosophy, University of Iowa, May 2014. Available at: https://ir.uiowa.edu/etd/4732/. 63, 64

RedHat. Red Hat + CentOS. Available at: https://community.redhat.com/centos-faq/, 2016. Online, Accessed: 01 March 2018. 226

T. C. Redman. Data's Credibility Problem. Harvard Business Review. Available at: https://hbr.org/2013/12/datas-credibility-problem, 2013. Online, Accessed: 23 March 2017. 118

R. H. Riffenburgh. *Statistics in Medicine*. Elsevier Science, 2011. ISBN 9780123848642. 49

L. Rokach and O. Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Company, 2014. ISBN 9789814590099. 62

M. Rouse. SearchSecurity - Secure Shell (SSH). Available at: https://searchsecurity.techtarget.com/definition/Secure-Shell, 2018. Online, Accessed: 14 April 2018. 229

N. Rozanski and E. Woods. *Software Systems Architecture – Working with Stakeholders Using Veiwpoints and Perspectives*. Addison-Wesley, Pearson Education, Inc. Rights and Contracts Department One Lake Street Upper Saddle

River, NJ 07458, 1st edition, 2005. ISBN 0321112296. 130, 131, 132, 133, 134, 135, 136

A. Ruckstuhl. Introduction to Nonlinear Regression. *ZHAW Zürcher Hochschule für Angewandte Wissenschaften*, 1(October), 2010. Available at:https://www.researchgate.net/publication/228785279_Introduction_to_Nonlinear_Regression. 49, 51, 55, 56

S. Rusitschka and A. Ramirez. Big Data Technologies and Infrastructures. (September):1–34, 2014. DOI: https://doi.org/10.5281/ZENODO.49166. 23, 24

P. Russom. Big data analytics. *TWDI Best Practices Report*, (Fourth Quarter): 1–34, 2011. ISSN 16113349. DOI: https://doi.org/10.1109/ICCICT.2012.6398180. 2, 11, 35

N. J. Salkind. *Encyclopedia of Measurement and Statistics*. Number v. 1 in A Sage reference publication. SAGE Publications, 2007. ISBN 9781412916110. 48, 49, 64

M. Saunders, P. Lewis, and A. Thornhill. *Research Methods for Business Students*. Pearson Education Ltd, Edinburgh Gate Harlow Essex CM20 2JE England, 2009. ISBN 978-0-273-71686-0. 6, 7

P. Schratz. Why you should use Linux for Data Science and R. Available at: https://pat-s.github.io/post/windows{_}vs{_}linux/, 2018. Online, Accessed: 02 March 2018. 153

Scikit-learn. Scikit-learn. http://scikit-learn.org/, 2017. Online, Accessed: 21 May 2018. 170, 171, 176, 177, 179, 181, 184, 260, 261, 262, 263, 264, 265

G. A. Seber and A. J. Lee. *Linear Regression Analysis*. Wiley-Interscience, Auckland, New Zealand, 2nd edition, 2003. ISBN 978-0-471-41540-4. 48, 50, 51, 52, 53, 54, 57, 60

U. Shafique and H. Qaiser. A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA). *Innovative Space of Scientific Research*, 12 (1):217–222, 2014. ISSN 2351-8014. 39

J. Shlens. A Tutorial on Principal Component Analysis. 2014. DOI: http://arxiv.org/abs/1404.1100. 100

M.-A. Sicilia. *Handbook of Metadata, Semantics and Ontologies*. World Scientific, 5 Toh Tuck Link, Singapore 596224, 2014. ISBN 9789812836298. 115

J. Sivakumar and K. S. Ravichandran. A Review on Semantic-Based Web Mining and its Applications. *International Journal of Engineering and Technology*, 5 (1):186–192, 2013. ISSN 0975-4024. 115, 116

L. I. Smith. A tutorial on Principal Components Analysis Introduction. Available at: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002. Online, Accessed: 10 June 2018. 100

A. Spark. Apache Spark. Available at: http://spark.apache.org/, 2018. Online, Accessed: 01 February 2018. 110, 111, 156, 164, 170, 171, 175, 177, 178, 179, 181, 184, 256, 258, 259, 265

D. Srivastava and X. Dong. Big data integration. *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1245–1248, 2013. ISSN 1063-6382. DOI: https://doi.org/10.1109/ICDE.2013.6544914. 19

SSH. SSH Protocol. Available at: https://www.ssh.com/ssh/protocol/, 2017. Online, Accessed: 06 June 2018. 229

S. Staab and R. Struder. *Handbook on Ontologies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2nd edition, 2009. ISBN 9783540709992. DOI: https://doi.org/10.1007/978-3-540-92673-3. 114, 116, 117

Stackoverflow. What is the difference between labeled and unlabeled data? Available at: https://stackoverflow.com/questions/19170603/what-is-the-difference-between-labeled-and-unlabeled-data?utm{_}medium=organic{&}autm{_}source=google{_}rich{_}qa{&}utm{_}campaign=google{_}rich{_}qa, 2013. Online, Accessed: 21 May 2018. 175

R. Steynberg. *Framework for identifying the most likely to be successful underprivileged tertiary bursary applicants*. M. Eng, Stellenbosch University, 2016. 62

Stratfor. How Many Countries Are There in the World in 2018. Available at: https://worldview.stratfor.com/article/how-many-countries-are-there-world-2018, 2018. Online, Accessed: 29 June 2018. 185

X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009(Section 3):1–19, 2009. ISSN 1687-7470. DOI: https://doi.org/10.1155/2009/421425. 97

D. K. Taft. Linux Is the Best OS for Big Data Apps: 10 Reasons Why. Available at: http://www.eweek.com/database/linux-is-the-best-os-for-big-data-apps-10-reasons-why, 2013. Online, Accessed: 18 February 2018. 153

S. Talegaon. Analytics of big data. 3(X):1124–1128, 2014. ISSN 2320-0790. 44

R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture: The Foundations, Theory, and Practice*. Wiley-Interscience, Danvers, MA USA, 1st edition, 2010. ISBN 978-0-470-16774-8. 12, 13, 34, 133

F. Tekiner and J. A. Keane. Big Data Framework. *IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pages 1494–1499, 2013. ISSN 1062-922X. DOI: https://doi.org/10.1109/SMC.2013.258. 104, 105

G. J. Tellis. Modeling marketing mix. *Handbook of marketing research*, pages 506–522, 2006. Available at: https://www.semanticscholar.org/paper/Modeling-Marketing-Mix-Tellis/decf182c226a4d0a73adeed7ae0cf9c243fd9c95. 49

G. J. Tellis and T. Ambler. *The SAGE Handbook of Advertising*. SAGE Publications, 2007. ISBN 9781412918862. 49

D. Tomar and S. Agarwal. A survey on Data Mining approaches for Healthcare. *International Journal of Bio-Science and Bio-Technology*, 5(5):241–266, 2013. DOI: https://doi.org/10.14257/ijbsbt.2013.5.5.25. 63

S. Tong and E. Chang. Support vector machine active learning for image retrieval. *Proceedings of the ninth ACM international conference on Multimedia*, page 107, 2001. ISSN 1581133944. DOI: https://doi.org/10.1145/500156.500159. 73, 81, 179

USMA. USMA Working Group, Department of Industrial Engineering, 2017. 36

L. Van Der Maaten, E. Postma, and J. Van Den Herik. Dimensionality Reduction : A Comparative Review. *October*, pages 1–35, 2009. ISSN 1532-4435. Available at: http://www.uvt.nl/ticc. 99, 100

V. Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer New York, 1999. ISBN 9780387987804. 63

J. P. Verma, B. Patel, and A. Patel. Big data analysis: Recommendation system with hadoop framework. *Proceedings - 2015 IEEE International Conference on Computational Intelligence and Communication Technology*, pages 92–97, 2015. DOI: https://doi.org/10.1109/CICT.2015.86. 105

C. Wessells. Caesium-137: A Deadly Hazard. Available at: http://large.stanford.edu/courses/2012/ph241/wessells1/, 2012. Online, Accessed: 18 June 2018. 184

T. White. *Hadoop: The Definitive Guide*. O'Reilly, Sebastopol, CA, 3rd edition, 2012. ISBN 9781449311520. 103, 104, 105, 107, 109

Wikibooks. Structure of the Internet: IP addresses. Available at: https://en.wikibooks.org/wiki/A-level_Computing/AQA/Computer_Components,_The_Stored_Program_Concept_and_the_Internet/Structure_of_the_Internet/IP_addresses, 2018. Online, Accessed: 27 February 2018. 226

C. J. Wild and G. A. F. Seber. *Nonlinear Regression*. Wiley-Interscience, 1st edition, 2005. ISBN 9780471725312. DOI: https://doi.org/10.1002/0471725315. 51, 55, 56

R. S. Williamson, M. Sahani, and J. W. Pillow. The Equivalence of Information-Theoretic and Likelihood-Based Methods for Neural Dimensionality Reduction. 2015. DOI: https://doi.org/10.1371/journal.pcbi.1004141. 99

I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining*. Morgan Kaufmann, Hamilton, New Zealand, 4th edition, 2016. ISBN 9780128042915. 15, 17, 37, 42, 43, 69, 81, 97, 183

F. Xhafa, L. Barolli, A. Barolli, and P. Papajorgji. *Modeling and Processing for Next- Generation Big-Data Technologies*. Springer Berlin Heidelberg, New Jersey, NY, USA, 1st edition, 2015. ISBN 9783319091761. 29, 30, 136, 143

R. Xu. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. ISSN 1045-9227. DOI: https://doi.org/10.1109/TNN.2005.845141. 87, 88

L. Yang, S. Liu, S. Tsoka, and L. G. Papageorgiou. A regression tree approach using mathematical programming. *Expert Systems with Applications*, 78:347–357, 2017. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.02.013. 48

S. Yin and O. Kaynak. Big Data for Modern Industry: Challenges and Trends. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7067026, 2015. Online, Accessed: 20 March 2017. 124

M. Zaharia, N. M. M. Chowdhury, M. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. (UCB/EECS-2010-53), May 2010. Available at: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-53.html. 110, 111

M. Zaharia, M. Chowdhury, T. Das, A. Dave, M. Justin, M. Mccauley, M. J. Franklin, S. Schenker, and I. Stoica. Fast and Interactive Analytics over Hadoop Data with Spark. Available at: https://www.usenix.org/system/files/login/articles/zaharia.pdf, 2012. Online, Accessed: 02 March 2018. 110, 111

L. Zhang, a. Stoffel, M. Behrisch, and S. Mittelstädt. Visual Analytics for the Big Data EraA Comparative Review of State-of-the-Art Commercial Systems. *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 173–182, 2012. ISSN 2325-9442. DOI: https://doi.org/10.1109/VAST.2012.6400554. 72

P. C. Zikopoulos, D. DeRoos, K. Parasuraman, T. Deutsch, D. Corrigan, and J. Giles. *Harness the Power of Big Data*. 2012. ISBN 978-0-07180818-7. 2, 3, 4, 11, 22

Appendix A

The process to configure Hadoop and Spark on a multinode cluster

The process to install and configure Hadoop and Spark on a multinode cluster is provided. The first step is to connect the nodes to a network (public or local), then check the connection to-and-from the different nodes and create a secure remote connection to the nodes. Thereafter, Hadoop and Spark were installed and configured. Due to limited resources and spare capacity (CPU and RAM) on the master node, the master node was configured to provide storage and processing as well, therefore there can be noted the addition of a third DataNode. This means that for this multinode configuration, there were three DataNodes and one NameNode, which served as the third DataNode or slave node (because of limited resources and available computing capacity).

A.1 Connecting the computers together on the network

A large component of any Big Data Analytics environment is the process of connecting the different nodes to one another, in order to distribute the storage and processing of information. This enables an analyst to leverage the power of distributed computers conducting processing in parallel to perform large computations at a faster rate than using just a single computer. The steps in order

to connect the three computers used in this project are outlined next, providing details of how to perform troubleshooting, and what to take into account during each step.

The steps outlined are for establishing a connection between computers using the *Linux* CENTOS 7 distribution. The steps remain the same regardless of the distribution chosen, the major differences are the package managers that are used among distributions. Package managers are tools used in *Linux* to find, install, update and manage software and software packages on the computers (Bearnes, 2018). For CENTOS, the package manager is called *yum* which comes from the *Red Hat* Linux distributions (RedHat, 2016), which CENTOS 7 is part of.

All steps regarding the input from a user in order to connect the computers to the network, install software *etc.* are done in a *terminal window* (an interface from which commands are executed, from installing, managing and running of programs and OS), if not,this will be specified at the relevant step in this and the following sections of this chapter.

The following steps are to be done after all nodes are connected together (over a network or locally) in a terminal window. To connect the computers together on a network, the steps are:

1. Identifying the IPv4 addresses (Internet Protocol version 4) of the different computers. An IP is the identifier of a computer on a network. To find the IP of a CENTOS 7 computer, the following command should be done on all nodes:

```
[hadoop@master]$ ifconfig
```

This will give the following output shown in Figure A.1, where the master node's IP is 146.232.144.48. The first six numbers are known as the *network* which indicate the network the computer is connected to (Wikibooks, 2018) and the last six numbers are known as the *host*, which identify the computer or node. Further information on IPs are given by Wikibooks (2018), discussing the difference between public and private IPs as well. The computers are connected to the university network, therefore the public IP with the 146.232 *network* codes were used, and not the local IPs of the nodes

given by the 192.168.

```
[hadoop@master ~]$ ifconfig
em1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 146.232.144.48  netmask 255.255.252.0  broadcast 146.232.147.255
        ether 34:17:eb:c2:14:aa  txqueuelen 1000  (Ethernet)
        RX packets 13368  bytes 2409525 (2.2 MiB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 3431  bytes 1638949 (1.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xf7100000-f7120000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1  (Local Loopback)
        RX packets 1486  bytes 246892 (241.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1486  bytes 246892 (241.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:90:8c:3c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure A.1: Running the *ifconfig* command to identify a computer's IP. The university network IP is given by the IP underlined in red, while the private/local IP of the computer is shown underlined in green.

The output of this command is shown in Figure A.1, which is specific to the network the computer is connected to. The IP's of the other nodes are 146.232.146.118 and 146.232.146.123 respectively. These IP's are used to connect all the nodes together in a cluster, discussed further in sections A.2 and A.3. The goal is to configure the data storage and analytic software on the three nodes, using these to identify them on the university network. To ensure the IPs do not change from the one originally assigned, 'static' IPs can be created, following the guide by Linglom.com (2017) or any other relevant source.

2. After identifying the various IPs of the selected nodes, the connection between the slave nodes and the master node need to be tested. Assuming

the computers are connected, in the terminal window, run

```
[hadoop@master]$ ping nodeA
```

from the master node for each of the slave nodes. The *nodeA* is the IP of the nodes. If a connection between the master node and slave nodes exist, the results should resemble the following given in Figure A.2. The results show information packets were sent and received between the master node and slave node, and therefore have the ability to send and receive data through the network (but not without a password).

It is assumed that all the computers used have CENTOS 7 installed, and each is given a unique *hostname* (a name given to a computer, used in the place of the IP) and user accounts. To change the hostname of a node, the following can be executed, *hostnamectl set-hostname NAME*, where *NAME* is the newly desired hostname. For this project, all nodes were given a common user account called *hadoop* and the hostnames were:

Hostnames:

- Master node: *master*
- Slave node 1: *datanode1*
- Slave node 2: *datanode2*

In Figure A.2, an example of the use of the user accounts (*hadoop*) and hostname are shown by the *hadoop@master* underlined in red, indicating the account and computer currently in use.

The next step involves securing the connection between the master and slave nodes, and establishing a password-less connection between nodes to allow easy data transfer.

A.1.1 Securing the connection

To establish a password-less connection between the master and slave node, the following steps can be followed. The goal is to ensure the nodes are able to share information remotely among one another. The method used to create this

```
[hadoop@master ~]$ ping datanode1
PING datanode1 (146.232.146.118) 56(84) bytes of data.
64 bytes from datanode1 (146.232.146.118): icmp_seq=1 ttl=64 time=16.3 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=2 ttl=64 time=16.0 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=3 ttl=64 time=12.5 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=4 ttl=64 time=15.1 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=5 ttl=64 time=15.9 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=6 ttl=64 time=16.6 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=7 ttl=64 time=16.0 ms
64 bytes from datanode1 (146.232.146.118): icmp_seq=8 ttl=64 time=19.5 ms
^C
--- datanode1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7009ms
rtt min/avg/max/mdev = 12.547/16.033/19.520/1.795 ms
```

Figure A.2: The results from running the *ping* command on a slave node, which show packets of information being sent (from the master node to the slave node) and received.

connection for this project is known as Secure Shell (SSH). Further information on the technology SSH is given by Rouse (2018) and SSH (2017). This method of encryption is not the only method available, but was chosen because it is used by most Hadoop-based Big Data installations, as is discussed in section A.2.

To create the SSH key that is to be shared amongst the three nodes, the following needed to be executed from the master node, using the guides from Rahul (2013), DWBI (2016), Fibrevillage (2016) and Darji (2017):

```
[hadoop@master]$ su root
[hadoop@master]# ssh-keygen -t rsa (at this point press enter until the key is generated, not entering any characters)
[hadoop@master]# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

The first step grants permissions to allow changes to be made, which is not permitted by the user account *hadoop*. Thereafter the key is created and then stored in an *authorized_keys* file. After these steps are complete, the SSH key needs to be copied amongst the slave nodes, until all nodes share the same keys.

From the master node, for all nodes, the following is to be executed for all three nodes:

```
[hadoop@master]$ ssh-copy-id -i ~/.ssh/authorized_keys account@hostname
(repeat for all nodes)
[hadoop@master]$ chmod 600 ~/.ssh/authorized_keys (gives permissions to the file)
[hadoop@master]$ chmod 700 ~/.ssh
```

The first step copies the key from the *authorized_keys* file to the nodes (*account@hostname*); the second and third steps grant the user (*hadoop*) permission to access the file (therefore the keys stored within). To check that all nodes have the same SSH keys, run the following in the terminal window on all nodes:

```
[hadoop@master]$ cat ~/.ssh/authorized_keys
```

If all keys are located in each of the nodes 'authorized_keys' file, the connection from the master node to the slave nodes can be tested:

```
[hadoop@master]$ ssh account@hostname
```

Running this command then gives the output shown in Figure A.3 indicating that a password-less and secure remote access between the node has been made, which allows for data transfers. This can then be done with all nodes (including the master) to ensure a successful connection.

```
[hadoop@master ~]$ ssh hadoop@datanode1
Last login: Tue Feb 27 18:29:25 2018 from master
[hadoop@datanode1 ~]$
```

Figure A.3: When logging into the various nodes, the 'last login' message serves to indicate a successful password-less remote access to the nodes, in this example successfully connecting to the slave node *datanode1*.

If there is a connection issue, check the firewall of the network using

```
[hadoop@master]$ su root
[hadoop@master]# firewall-cmd --get-active-zones
```

To allow access on the network between the nodes, open a common port for all nodes to use on the firewall, by running

```
[hadoop@master]# firewall-cmd --zone=public --add-port=NUMBER/tcp
--permanent
```

and then,

```
[hadoop@master]# firewall-cmd --reload
```

The NUMBER refers to a port (number) selected, and zone is set to public. Using a unique port ensures a dedicated connection between nodes. To prevent possible port issues, port 22 was used in this project, as it is the default SSH port used to send and receive data.

Next, the steps that were used to install the storage system for this project are outlined, along with reasons as to why this particular system was used.

A.2 Data storage solution

The steps followed in order to install and configure the storage system were a combination of the steps from Darji (2017), Fibrevillage (2016), DWBI (2016) and Rahul (2013). These steps were carried out after the three nodes were connected to the university network, and SSH connections were established:

Firstly, installing the necessary Java environment, at the time of this project, Java version 8 (1.8.0_161) was used:

```
[hadoop@master]$ su root (accessing root computer, to exit type exit)
[hadoop@master]# yum upgrade
```

If this did not install the necessary Java environment required by Hadoop, the guide from Rahul (2016) can be used or any other relevant guides that can be found online. After this, on all nodes, the IPs and hostnames need to be provided, such that the Hadoop software is aware of the computers being used.

Run the following on each node:

```
[hadoop@master]$ su root
[hadoop@master]# nano /etc/hosts
```

Enter the IP and hostnames as shown in Figure A.4. Now the Hadoop software can be downloaded, installed and configured to provide the storage capability for this project. The latest Hadoop version at the time of this project was hadoop-2.8.3 and that was installed on the *master* node. The following steps should be carried out on the master node, the first nine steps should also be conducted on the slave nodes:

```
  GNU nano 2.3.1              File: /etc/hosts                          Modified

<!--
#127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
#::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
-->

146.232.144.48 master
146.232.146.118 datanode1
146.232.146.123 datanode2
```

Figure A.4: How the IP addresses and hostnames need to be added to the *etc/hosts* file in each node, for this project the master node along with the two slave nodes are appended to the file.

```
[hadoop@master]$ su root
[hadoop@master]# mkdir -p /opt/hadoop
[hadoop@master]# mkdir -p /home/volume/namenode
[hadoop@master]# mkdir -p /home/volume/namesecondary
[hadoop@master]# mkdir -p /home/volume/datanode
[hadoop@master]# mkdir -p /home/volume/yarn/local
[hadoop@master]# mkdir -p /home/volume/yarn/log
[hadoop@master]# chmod -R 755 /opt/hadoop (do this for all other newly created
directories)
[hadoop@master]# chown -R hadoop:hadoop /opt/hadoop (do this for all other
newly created directories, then exit root)
[hadoop@master]$ cd /opt/hadoop (changing to the directory)
[hadoop@master hadoop]$ wget http://www.us.apache.org/dist/hadoop/common
/hadoop-2.8.3/hadoop-2.8.3.tar.gz
```

```
[hadoop@master hadoop]$ tar -xzvf hadoop-2.8.3.tar.gz
```

The first step is to access the root of the computer, create the directories required by Hadoop, and grant read and write permissions to these directories (*chmod* commands). As discussed in section 2.4.1, Hadoop makes use of NameNodes, SecondaryNodes (backup storage of the Metadata stored in NameNode) and DataNodes (nodes which store the data) and have to be given relevant directories. Thereafter using the *chown* command, access for the *hadoop* user is provided. The compressed Hadoop installation file is then downloaded to the '/opt/hadoop' directory and uncompressed using the *tar* command.

After this, the paths to and from files, programs *etc.* need to be provided, which Hadoop uses to access these files and programs. To do so, in the terminal window access the *.bashrc* file by typing,

```
[hadoop@master]$ nano ~/.bashrc
```

and in the file add:

```
export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk (path where java is located
on the computers used in this project)
export PATH=$PATH:$JAVA_HOME/bin
HADOOP_HOME=/opt/hadoop/hadoop-2.8.3 (path where hadoop is stored)
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.
export HADOOP_OPTS="$HADOOP_OPTS -Djava.security.egd=
file:/dev/../dev/urandom"
```

After these have been appended to the file, for it to take effect execute

```
[hadoop@master]$ source ~/.bashrc
```

To confirm that Hadoop is installed the following can be run, *hadoop version*, which should give the following output shown in Figure A.5.

```
[hadoop@master ~]$ hadoop version
Hadoop 2.8.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r b3f
e56402d908019d99af1f1f4fc65cb1d1436a2
Compiled by jdu on 2017-12-05T03:43Z
Compiled with protoc 2.5.0
From source with checksum 9ff4856d824e983fa510d3f843e3f19d
This command was run using /opt/hadoop/hadoop-2.8.3/share/hadoop/com
mon/hadoop-common-2.8.3.jar
```

Figure A.5: The output provided when checking the Hadoop version, indicating Hadoop-2.8.3 version is installed.

Next, Hadoop needs to be built for the specific configuration of this project. The configuration files used in the next steps of the Hadoop installation, are provided in section A.2.1. The following are the steps that need to be followed in order to do so. First navigate to the directory where the configuration files are located, by using,

```
[hadoop@master]$ cd $HADOOP_HOME/etc/hadoop/
```

The first file, *core-site.xml* is modified, by typing

```
[hadoop@master hadoop]$ nano core-site.xml
```

and entering the properties given in Figure A.9, into the file. The configuration file code should be entered between <*configuration*> and </*configuration*>, already present in the *core-site.xml* file. The same steps are followed with the *hdfs-site.xml* file, the configuration code shown in Figure A.10, *mapred-site.xml*'s file, with configuration code shown in Figure A.11 and *yarn-site.xml* file, the configuration code shown in Figure A.12. Then the path to where Java is located on the nodes need to be provided. This is done in the *hadoop-env.sh* file, entering

```
JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk (Java location for these nodes)
```

in the *JAVA_HOME* location. Next, the NameNode and DataNodes need to be specified. In the same directory, the *master* and *slaves* files need to be changed, by typing

```
[hadoop@master hadoop]$ nano masters
```

and entering the hostname (*master*) into the file. Within the *slaves* file, typing

```
[hadoop@master hadoop]$ nano slaves
```

and entering the DataNodes hostnames, *master* , *datanode1* and *datanode2* for this project. The NameNode is to be configured after this step, by executing the following command,

```
[hadoop@master hadoop]$ hdfs namenode -format
```

which formats the NameNode and configures the storage directories specified in the configuration files previously completed. The output after a successful formatting, shown in Figure A.6 indicates the NameNode is successfully formatted. Another indication of successful configuration is giving by the "Exiting with status 0" message of 0, which indicates successful formatting. An unsuccessful formatting would give a 1 instead of 0.

```
19749084399
18/02/27 18:31:24 INFO common.Storage: Storage directory /home/volume/namenode has been successfull
y formatted.
18/02/27 18:31:24 INFO namenode.FSImageFormatProtobuf: Saving image file /home/volume/namenode/curr
ent/fsimage.ckpt_0000000000000000000 using no compression
18/02/27 18:31:24 INFO namenode.FSImageFormatProtobuf: Image file /home/volume/namenode/current/fsi
mage.ckpt_0000000000000000000 of size 323 bytes saved in 0 seconds.
18/02/27 18:31:24 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
18/02/27 18:31:24 INFO util.ExitUtil: Exiting with status 0
18/02/27 18:31:24 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at master/146.232.144.48
************************************************************/
```

Figure A.6: The output message indicating the NameNode is successfully formatted.

The slave nodes that have to serve as the DataNodes are configured after the successful formatting of the NameNode. The complete Hadoop directory is to be copied to all DataNodes in the cluster,

```
[hadoop@master]$ cd /opt
[hadoop@master opt]$ scp -r hadoop hadoop@datanode1:/opt
[hadoop@master opt]$ scp -r hadoop hadoop@datanode2:/opt
```

Similar to the master node, the ∼/.bashrc file in each of the slave nodes (DataNodes) need to be updated. Copy the contents of the file from the master node and login to each of the slave nodes, copy the contents into the slave nodes' ∼/.bashrc file. For illustrative purposes,

```
[hadoop@master]$ ssh hadoop@datanode1
[hadoop@datanode1]$ nano ∼/.bashrc (and then copy the
contents into the file)
[hadoop@datanode1]$ source ∼/.bashrc
[hadoop@datanode1]$ exit
```

and then for the second slave node the same process is followed,

```
[hadoop@master]$ ssh hadoop@datanode2
[hadoop@datanode2]$ nano ∼/.bashrc (and then copy the
contents into the file)
[hadoop@datanode2]$ source ∼/.bashrc
[hadoop@datanode2]$ exit
```

Now that all the nodes have the required files, directories and each node correctly points (from the ∼/.bashrc file) to the files and directories, the Hadoop cluster can be started from the master node (NameNode).

```
[hadoop@master]$ $HADOOP_HOME/sbin/start-dfs.sh
[hadoop@master]$ $HADOOP_HOME/sbin/start-yarn.sh
```

The configuration of Hadoop was done successfully, giving the output shown in Figures A.7 and A.8, when the software is started.

```
[hadoop@master hadoop]$ $HADOOP_HOME/sbin/start-dfs.sh
Starting namenodes on [master]
master: starting namenode, logging to /opt/hadoop/hadoop-2.8.3/logs/hadoop-hadoop-namenode-master.o
ut
master: starting datanode, logging to /opt/hadoop/hadoop-2.8.3/logs/hadoop-hadoop-datanode-master.o
ut
datanode1: starting datanode, logging to /opt/hadoop/hadoop-2.8.3/logs/hadoop-hadoop-datanode-datan
ode1.out
datanode2: starting datanode, logging to /opt/hadoop/hadoop-2.8.3/logs/hadoop-hadoop-datanode-datan
ode2.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/hadoop-2.8.3/logs/hadoop-hadoop-seconda
rynamenode-master.out
```

Figure A.7: The output given when the Hadoop services are started.

```
[hadoop@master hadoop]$ $HADOOP_HOME/sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /opt/hadoop/hadoop-2.8.3/logs/yarn-hadoop-resourcemanager-mast
er.out
master: starting nodemanager, logging to /opt/hadoop/hadoop-2.8.3/logs/yarn-hadoop-nodemanager-mast
er.out
datanode2: starting nodemanager, logging to /opt/hadoop/hadoop-2.8.3/logs/yarn-hadoop-nodemanager-d
atanode2.out
datanode1: starting nodemanager, logging to /opt/hadoop/hadoop-2.8.3/logs/yarn-hadoop-nodemanager-d
atanode1.out
```

Figure A.8: The output given when the Hadoop YARN services are started.

To check the configured capacity of the newly created Hadoop HDFS database, the following pages can be opened in a web-browser providing a dashboard of various statistics of the Hadoop cluster created:

- *master:50070* or using the IP, *146.232.144.48:50070* - The URL provided to check the status of the NameNode and configured capacity of the system, shown in Figure A.13.
- *master:50075* or using the IP, *146.232.144.48:50075* - The URL provided to check the status of the DataNodes, shown in Figure A.14.
- *datanode1:50075* or using the IP, *146.232.146.118:50075* - The URL provided to check the status of the DataNodes, shown in Figure A.15.
- *datanode2:50075* or using the IP, *146.232.146.123:50075* - The URL provided to check the status of the DataNodes, shown in Figure A.16.

The hadoop configuration files that were used are given in the next section A.2.1.

A.2.1 The Hadoop configuration files

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://master:8020/</value>
</property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
</property>
</configuration>
```

Figure A.9: The configuration file *core-site.xml*, used in the Hadoop installation process. The file sets the NameNode name according the hostname of the master node and defaults to a 128MB block size for data replication.

```
<property>
<name>dfs.namenode.name.dir</name>
<value>/home/volume/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/volume/datanode</value>
</property>
<property>
<name>dfs.namenode.checkpoint.dir</name>
<value>/home/volume/namesecondary</value>
</property>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>dfs.block.size</name>
<value>134217728</value>
</property>
```

Figure A.10: The configuration file *hdfs-site.xml*, used in the Hadoop installation process. The path to the NameNodes and DataNodes are specified, along with the secondary NameNode (a backup of the metadata stored in NameNode) and finally, specifying the number of times data is replicated accrosss the system.

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobhistory.address</name>
<value>master:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>master:19888</value>
</property>
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/user/app</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Djava.security.egd=file:/dev/../dev/urandom</value>
</property>
</configuration>
```

Figure A.11: The configuration file *mapred-site.xml*, used in the Hadoop installation process. In this file the necessary directories and paths for the Hadoop framework to conduct MapReduce tasks is specified.

```
<property>
<name>yarn.resourcemanager.hostname</name>
<value>master</value>
</property>
<property>
<name>yarn.resourcemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<property>
<name>yarn.nodemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.log-aggregation-enable</name>
<value>true</value>
</property>
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>/home/volume/yarn/local</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>/home/volume/yarn/log</value>
</property>
<property>
<name>yarn.nodemanager.remote-app-log-dir</name>
<value>hdfs://master:8020/var/log/hadoop-yarn/apps</value>
</property>
```

Figure A.12: The configuration file *yarn-site.xml*, used in the Hadoop installation process. In this file, the relevant paths and directories are specified for yarn related tasks, required by Hadoop.

The various Hadoop dashboards that are provided to give the user insights into the state of the system, from configured capacity to the nodes that are operational *etc.*

A.2.2 The Hadoop dashboards

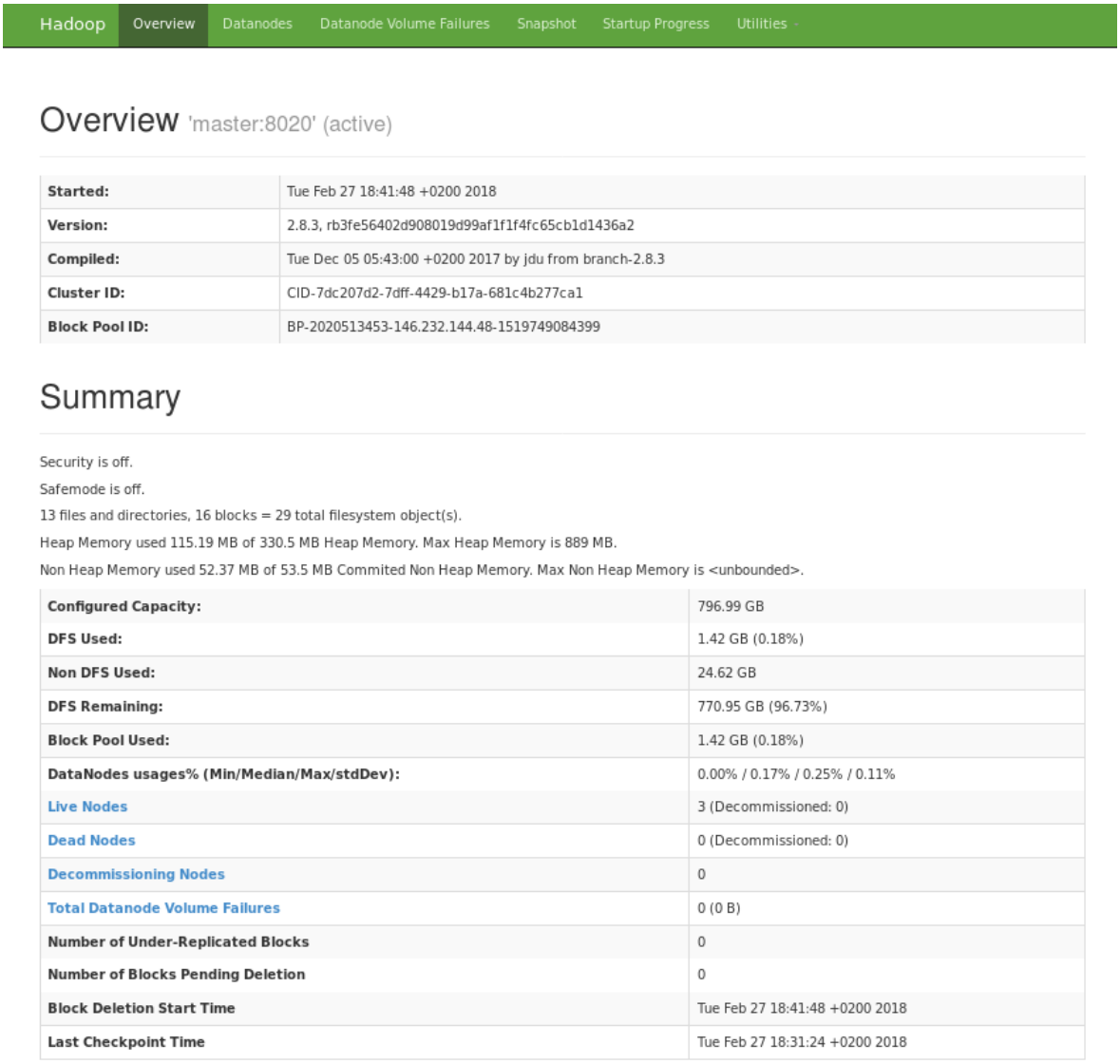


Figure A.13: The dashboard provided on the NameNode, indicating the configured capacity of the Hadoop HDFS cluster, the number of nodes connected *etc.*

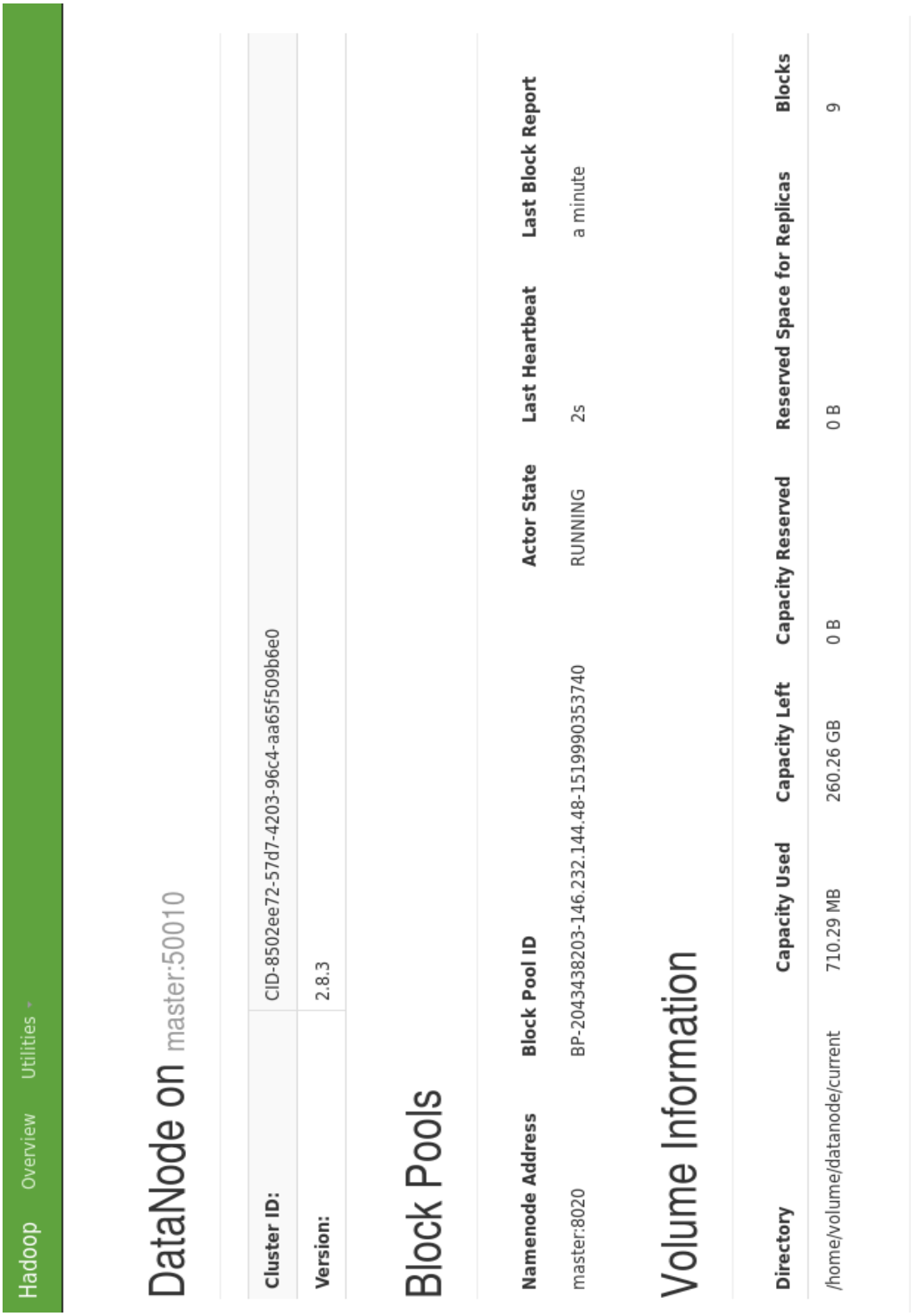


Figure A.14: The dashboard provided on the third DataNode configured on the master node (*master*), indicating the configured capacity, unique ID *etc.*

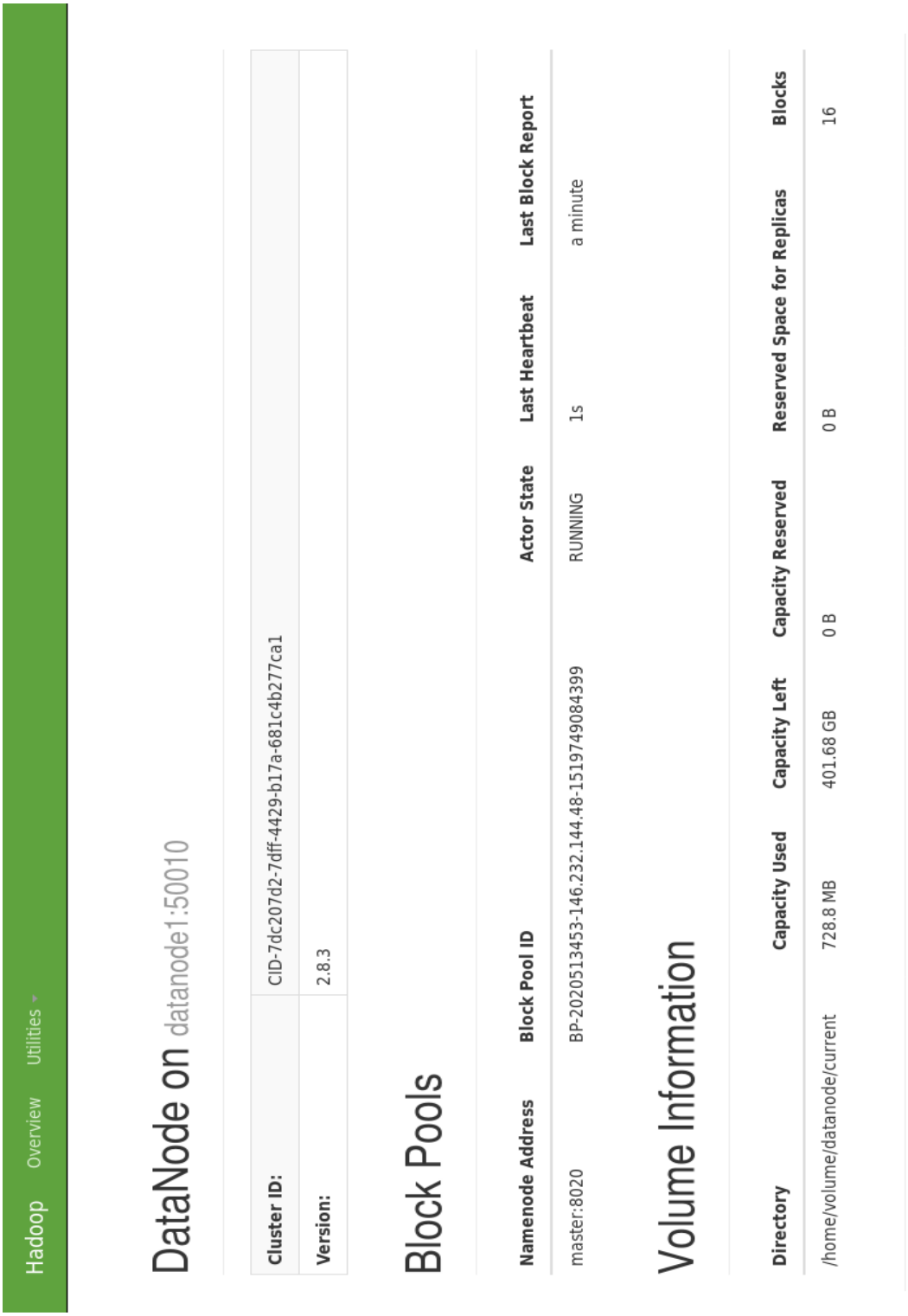


Figure A.15: The dashboard provided on the first DataNode (*datanode1*), indicating the configured capacity, unique ID *etc.*

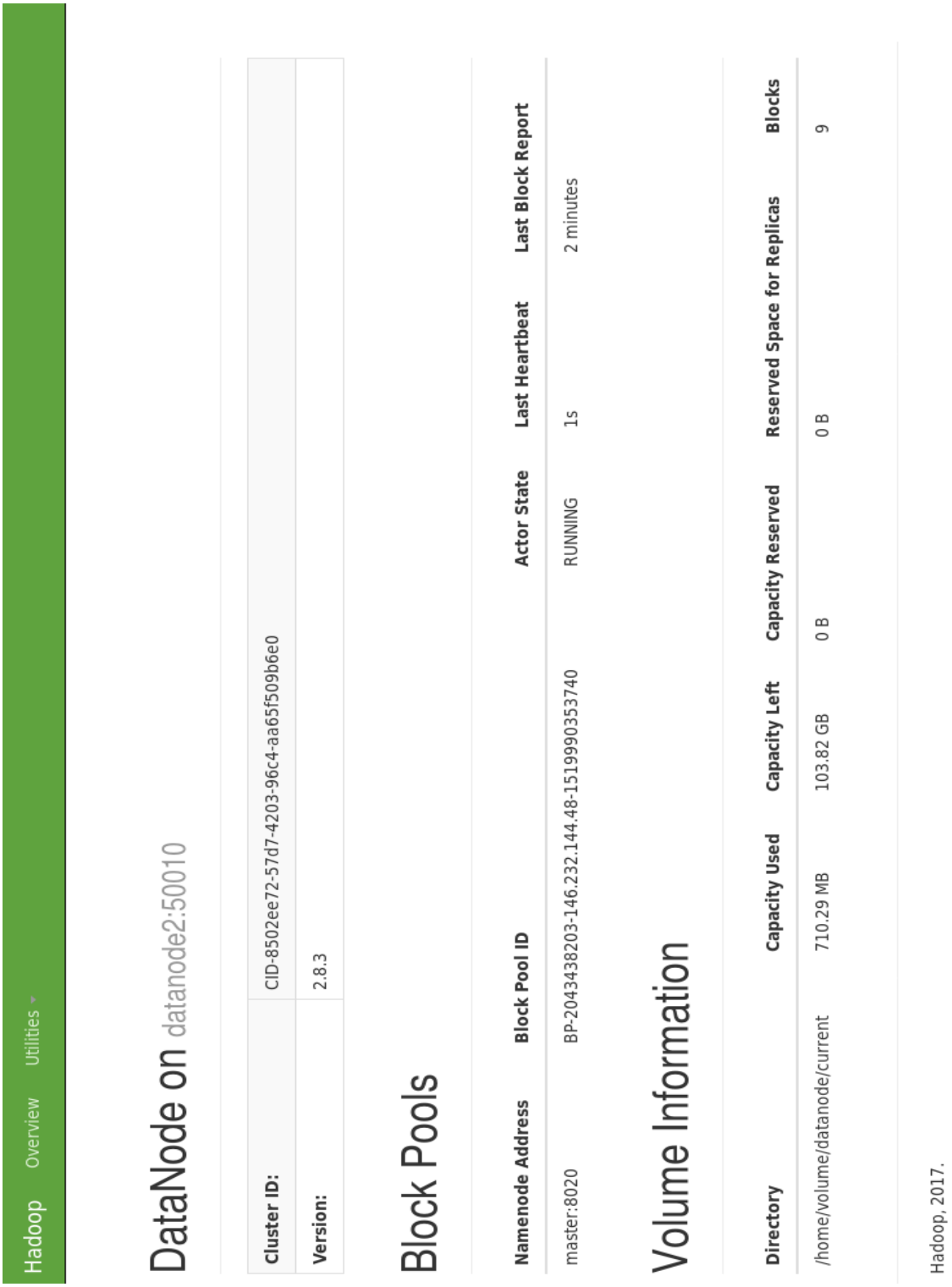


Figure A.16: The dashboard provided on the second DataNode (*datanode2*), indicating the configured capacity, unique ID *etc.*

A.2.3 Working with the HDFS environment

The following outlines how files are added or removed from the HDFS environment, directories made and finally how to access the files in order to start a query, using Spark to analyse.

To add files to the HDFS environment, the following is run in a terminal window:

```
[hadoop@master]$ hdfs dfs -put /location/of/file.py
/location/to/store/in/hadoop
```

The first part calls the HDFS environment, whereby the next informs a file is to be added, then specifying the files local location, and then where in the HDFS environment it should be stored.

To remove a file in the HDFS environment, the following can then be run, which calls the HDFS environment and then the remove function, then specifying the location of the file:

```
[hadoop@master]$ hdfs dfs -rm /File/Location/
```

To specify a directory in which to store a file, the following can then be executed, calling the HDFS environment, thereafter the create directory function and specifying the depth:

```
[hadoop@master]$ hdfs dfs -mkdir /Directory/Name
```

In the next section A.3, the various steps that were followed to install and configure the Spark analytics software used in this project, is discussed.

A.3 Data analytics solution

The steps that were followed to install and configure the Spark environment used in this project are provided. Spark is installed to provide the analytics system to analyse large datasets, making use of the built-in libraries and Python API provided. The steps to install and configure Spark were a combination of DeZyre (2018), DataDotz (2018) and Mitra (2016). These steps were carried out after

the three nodes were connected to the university network, and SSH connections were established. The installation steps are provided assuming Hadoop is already installed. The specific Spark version used in this project is Spark-2.8.3. As with the Hadoop installation, firstly, Java needs to be installed on all nodes (assuming it is not already installed from the Hadoop installation step). The specific Java version used at the time of this project was Java (1.8.0_161):

```
[hadoop@master]$ su root (accessing root computer, to exit type exit)
[hadoop@master]# yum upgrade
```

If this did not install the required Java environment, the guide by Rahul (2016) can be used, or other relevant sources found online.

The next step requires that a Spark directory be created and the necessary permissions are given such that the *hadoop* user can access the Spark files.

```
[hadoop@master]$ su root
[hadoop@master]# mkdir -p /opt/spark
[hadoop@master]# chmod -R 755 /opt/spark
[hadoop@master]# chown -R hadoop:hadoop /opt/spark
[hadoop@master]$ cd /opt/spark (changing to the directory)
[hadoop@master spark]$ wget http://www-us.apache.org/dist/spark/spark-2.2.1/spark-2.2.1-bin-hadoop2.7.tgz
[hadoop@master spark]$ tar -xzvf spark-2.2.1-bin-hadoop2.7.tgz
```

This step creates the Spark directory with the permissions, then the compressed Spark file containing the software is downloaded, and then uncompressed.

The next step appends the paths to the *.bashrc* file, such that when using Spark, the software is able to locate the relevant locations of the files.

```
[hadoop@master]$ nano ~/.bashrc
```

Append the following to the file to point to the required files. These additions need to be added to all nodes:

```
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
```

To complete the addition of these paths, the *source* command needs to be

executed:

```
[hadoop@master]$ source ~/.bashrc
```

The next step involves configuring the Spark environments, to point the Spark software to where the Java and Hadoop files are located. There also needs to be specified how many of the CPU processing cores are to be used by the Spark program (ensuring enough cores are left for basic functions of the nodes). These changes are made to the *spark-env.sh* file:

```
[hadoop@master]$ cd $SPARK_HOME/conf
[hadoop@master conf]$ cp spark-env.sh.template spark-env.sh
```

The *spark-env.sh* is first created, and can now be opened to add the specified requirements:

```
[hadoop@master]$ nano spark-env.sh
```

and then in the file, add

```
export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk
export HADOOP_CONF_DIR=/opt/hadoop/hadoop-2.8.3/etc/hadoop
export SPARK_WORKER_CORES=6 (using 6 of the 8 available cores available on the nodes CPU's)
```

The next step requires that the slave nodes be configured for Spark. Similar to Hadoop, the *slaves* file in the SPARK_HOME directory needs to be updated with *master, datanode1* and *datanode2*. The final step in the installation process before the Spark system can be started, is to send the Spark files created and installed on the NameNode to all the DataNodes in the system:

```
[hadoop@master]$ cd /opt
[hadoop@master opt]$ scp -r spark hadoop@datanode1:/opt
[hadoop@master opt]$ scp -r spark hadoop@datanode2:/opt
```

The final step after this process is complete is to start all Spark processes:

```
[hadoop@master]$ $SPARK_HOME/sbin/start-all.sh
```

The output to confirm the Spark installation was successful is given in Figure A.17, in the terminal window. The output indicates the various worker nodes have started. Then navigating to the Spark dashboards will allow confirmation of the successful installation and configuration of the Spark software, provided in section A.3.1. The first dashboard shown in Figure A.18 provides the user with the Spark configuration, from the number of CPU cores and RAM available on each worker node. The dashboard also provides a output of current and past applications being executed in Spark. An analyst can use this to monitor the state of a query to terminate a query if deemed necessary. If the analyst also wishes to deploy queries to the cluster, the dashboard provides the URL (*spark://master:7077*) that an analyst provides before running a query.

The second dashboard, shown in Figure A.19 provided by Spark, allows an analyst to monitor individual queries being run on the Spark cluster as a query is being executed. The query shown, for illustrative purposes, allows an analyst to see individual processes being run at various stages, and the duration of each stage and the query as a whole. The following URLs are used to access these dashboards on a web-browser:

- *master:8080*: The URL to navigate to the dashboard provided by Spark, shown in Figure A.18.
- *master:4040*: The URL to navigate to the dashboard provided by Spark, shown in Figure A.19.

```
[hadoop@master ~]$ $SPARK_HOME/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/spark/spark-2.2
.1-bin-hadoop2.7/logs/spark-hadoop-org.apache.spark.deploy.master.Master-1-maste
r.out
master: starting org.apache.spark.deploy.worker.Worker, logging to /home/spark/s
park-2.2.1-bin-hadoop2.7/logs/spark-hadoop-org.apache.spark.deploy.worker.Worker
-1-master.out
datanode2: starting org.apache.spark.deploy.worker.Worker, logging to /home/spar
k/spark-2.2.1-bin-hadoop2.7/logs/spark-hadoop-org.apache.spark.deploy.worker.Wor
ker-1-datanode2.out
datanode1: starting org.apache.spark.deploy.worker.Worker, logging to /home/spar
k/spark-2.2.1-bin-hadoop2.7/logs/spark-hadoop-org.apache.spark.deploy.worker.Wor
ker-1-datanode1.out
[hadoop@master ~]$
```

Figure A.17: The output message provided by Spark when starting the Spark software on all the nodes, indicating the worker nodes on each computing node in the cluster.

A.3.1 The Spark dashboards

In this section, the different dashboards are displayed that are provided by spark to monitor the workers configured in a cluster and the state of a query being executed.

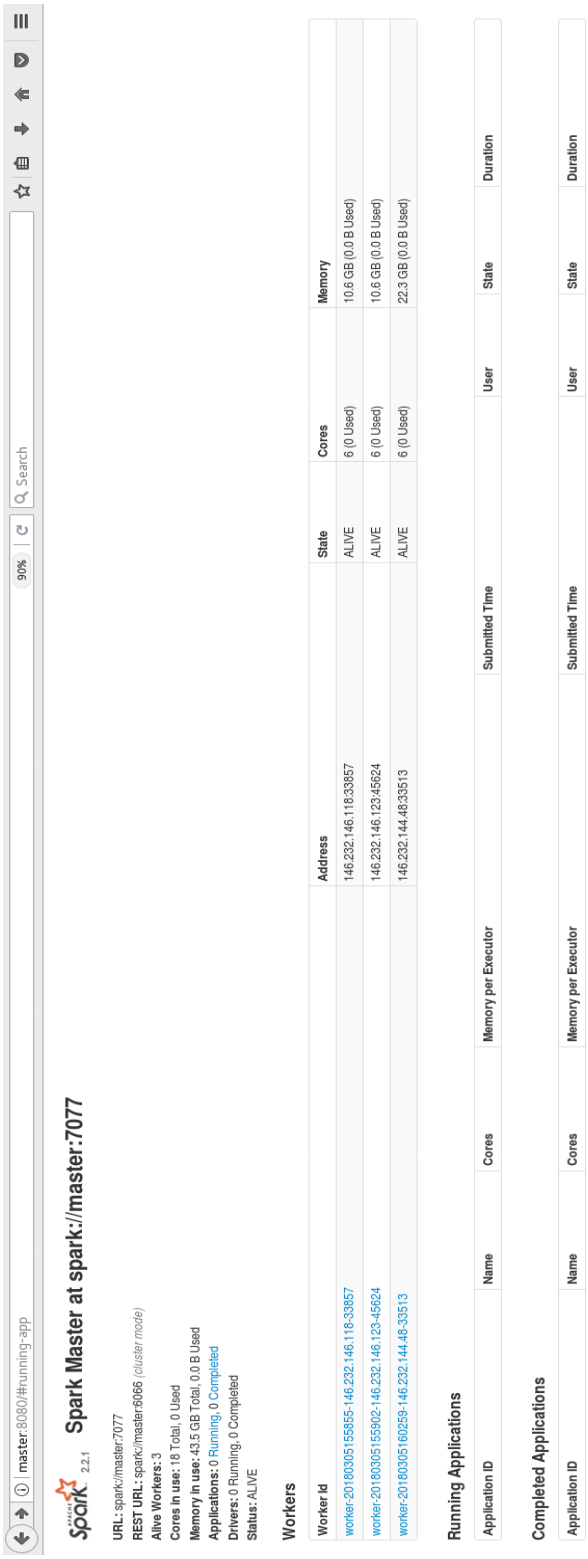


Figure A.18: The first dashboard provided to a user with the status of the Spark configuration, indicating the number of CPU cores and RAM configured for each worker node, and the current and past applications being executed.

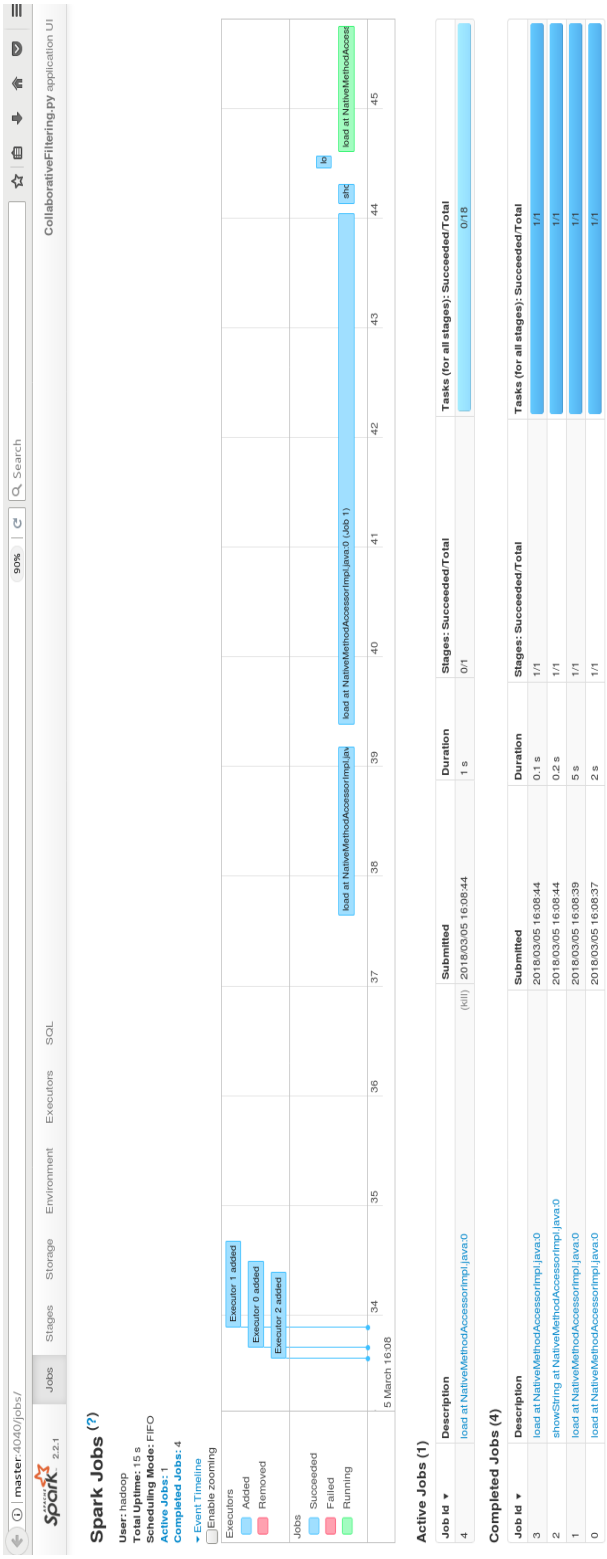


Figure A.19: A dashboard provided by Spark to monitor individual spark queries at various stages; the query shown is for illustrative purposes.

A.3.2 Creating the Spark environment

The following are different pieces of code required in Python to start a Spark environment and conduct queries on the cluster. This includes the PySpark package along with the required terms to start and stop the Spark environment. To start the environment the *'SparkSession'*, *'SparkConf'*, *'SparkContext'* are called first. Next, the cluster itself is configured and called using the URL master found on the Spark dashboard:

```
conf = SparkConf().setMaster("spark://master:7077)
sc = SparkContext(conf=conf)
```

The Spark environment is then customised, specifying different system configurations such as the program file to be run and the amount of RAM to be used by each node:

```
spark = SparkSession.builder.master("spark://master:7077)
.appName("NameOfFile")
.config("spark.executor.memory", "10gb").getOrCreate()
```

After adding these to the beginning of the file, to locate the relevant file stored in HDFS, and conduct an analysis the following can be followed,

```
VariablePath = "hdfs://master:8020/Location/Path/FileName.csv"
variableData = sql_sc.read.format('com.databricks.spark.csv')
.options("delimiter", ',').options(etc.).load(VariablePath)
```

This locates the file, and using the SQL package within Pyspark, provided the file is a .csv or a text file in this example, loads the file into a table, providing the relevant options to correctly read the file, which can be customised accordingly. Similar steps can be followed to load data from other formats, stored in HDFS. To close the Spark environment and complete the spark analysis, the following two statements are added at the end of the file:

```
spark.stop()
sc.stop()
```

To then run a query, using the file(s) stored in the HDFS environment, the

following can be run in a shell environment, assuming 10GB of RAM per executor and driver node is used, by calling the Python (.py) file which runs the query:

```
spark-submit --executor-memory 10G --driver-memory 10G
--master spark://master:7077 /Location/Of/The/Spark/Query/File.py
```

Appendix B

Literature of the algorithms used by Spark and Scikit-learn

The relevant literature is provided here for each of the ML algorithms used by the Spark (BDAD) and Scikit-learn (SS) analytics software and libraries in the validation phase in Chapter 5. This appendix provides background on the method by which each ML technique conducts an analysis. The following discussion is divided according to the different ML tools, under which the respective literature of each of the techniques used in this project is then discussed. To avoid repetition, if the literature overlaps between the Spark and Scikit-learn algorithms, only one detailed discussion will be given.

B.1 Regression machine learning techniques

The literature found on how the Spark and Scikit-learn algorithms perform different regression analysis is provided.

B.1.1 Logistic regression algorithm in Spark

Literature provided by Spark on logistic regression covers multinomial logistic regression, as this allows more labels or outcomes greater than two. The algorithm functions as follows, from Spark (2018).

There are K sets of coefficients and J number of features in a given dataset. The conditional probabilities of each outcomes are calculated by

$$P(Y = k|\boldsymbol{X}, \boldsymbol{\beta_k}, \beta_{0k}) = \frac{\exp^{\boldsymbol{\beta_k} \cdot \boldsymbol{X} + \beta_{0k}}}{\sum_{k'=0}^{K-1} \exp^{\boldsymbol{\beta_{k'}} \cdot \boldsymbol{X} + \beta_{0k'}}} \tag{B.1}$$

and the weighted negative log-likelihood is minimised, making use of an elastic-net penalty to manage the possibilities of overfitting. The regression coefficients are given by β, with k the outcome from the set K. This function is given by the first component including the log likelihood function, and the regularisation function or elastic-net penalty given after the λ, with λ being the regularisation parameter and α the elastic net parameter,

$$\min_{\beta_k \beta_0} - \left[\sum_{i=1}^{L} \omega_i \cdot \log P(Y = y_i|\boldsymbol{X_i}) \right] + \lambda \left[\frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||^2 + \alpha||\boldsymbol{\beta}||_1 \right]. \tag{B.2}$$

X denotes the feature set, with $\boldsymbol{X_i}$ a given feature within X. The term Y is the class or label outcomes, with y_i a specific outcome within the set of outcomes Y.

B.1.2 Logistic regression algorithm in Scikit-learn

As with the algorithm in Spark, the logistic regression algorithm implemented in Scikit-learn can be applied to multinomial datasets. The function also incorporates regularisation terms, but divides the function given binary classes or multinomial. When working with binomial classes, a L2 regularisation is added to the logistic regression function, given by

$$\min_{\omega, c} \frac{1}{2}\omega^T \omega + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T \omega + c)) + 1) \tag{B.3}$$

with the L2 regularisation term weights given by ω and regularisation coefficient parameter C. The L2 regularisation is incorporated to ensure the algorithm generalises better to unseen data (preventing overfitting). The definitions for the X and y terms are as used in the Spark discussion above. When conducting the multinomial classes, the L1 regularisation term is added to the logistic regression's log-likelihood function. This function is given by,

$$\min_{\omega,c}||\omega||_1 + C\sum_{i=1}^{n}\log(\exp(-y_i(X_i^T\omega + c)) + 1) \tag{B.4}$$

where the differentiating factor is in the L1 just adds the absolute value of the weights as a form of penalty Nagpal (2017). As stated in Nagpal (2017), L1 regularisation reduces the impact of features of less importance, which is required when working with multinomial classes to prevent overfitting (having the one predictor dominating the features).

B.1.3 Linear regression algorithm in Spark

As stated in Spark (2018) literature, the model for linear regression uses the same structure as used for the logistic regression model when conducting the analysis. The difference is, instead of the logistic function outlined in section 5.5.2, the Spark linear regression function uses the linear component found in (B.1),

$$\boldsymbol{\beta_k} \cdot \boldsymbol{X} + \beta_{0k} \tag{B.5}$$

to perform the LR analysis.

B.1.4 Linear regression algorithm in Scikit-learn

The Scikit-learn library performs linear regression using the same function as that used by Spark and shown in (5.1). For Scikit-learn, an *ordinary least squares* function is then used to measure or evaluate the prediction made by linear regression

$$\min_{w} ||X_w - y||^2{}_2, \tag{B.6}$$

with w the coefficients of the dataset X and y the predicted value.

B.2 Classification machine learning techniques

The following discussion includes literature for the various Classification algorithms found in Spark and Scikit-learn and used in this project namely, decision trees, SVM, naive bayes and multilayer perceptron classifier (a type of ANN).

B.2.1 Decision tree algorithm in Spark

The following description of the decision tree algorithm is from Spark (2018). As stated, the algorithm performs recursive binary partitioning of the feature space in a greedy fashion, by choosing the best split from a set of possible splits. The goal is to maximise the information gain at a node of a tree

$$\arg\max_{s} IG(D, s)$$

when a split (s) occurs within the dataset D. Spark checks the state or quality of the split at a node using two measures, the *Gini impurity* and *Entropy*. A third, *Variance* is used for decision tree regression, therefore is not discussed here, as it is not used in this project (reading on this measure can be found at Spark (2018)).

The *Gini impurity* measures the frequency of a label (i) at a node (f_i), with C unique labels, in order to determine a split, this is given by

$$\sum_{i=1}^{C} f_i(1 - f_i).$$

For the *Entropy* measure, the same strategy is employed, but is done using the following equation,

$$\sum_{i=1}^{C} -f_i \log(f_i).$$

The information gain (IG) between a parent node and the weighted sum of two child nodes, using these measures are then provided as follows,

$$IG(D, s) = \text{Impurity}(D) - \frac{N_{left}}{N}\text{Impurity}(D_{left}) - \frac{N_{right}}{N}\text{Impurity}(D_{right}) \tag{B.7}$$

where a node split partitions the dataset D of size N into two datasets ($left$ and $right$). The $Impurity$ is where a either the *Gini* or *Entropy* impurity functions is inserted in (B.7) to measure the quality of split. The Spark literature also provides the following three stopping conditions of the decision tree algorithm:

1. When the node depth is equal to that of the maximum depth provided,

2. Possible splits do not lead to an information gain (IG) that is larger than the minimum information gain, and finally,

3. Possible splits do not provide child nodes that have a minimum number of instances following from that child node.

B.2.2 Decision tree algorithm in Scikit-learn

The Decision Tree algorithm implemented by Scikit-learn uses the CART algorithm with a greedy approach. The following is from Scikit-learn (2017), describing the mathematical formulae used in order to develop the decision tree algorithm, using the same symbols as in section B.2.1. The discussion only includes the formulae used for decision tree Classifiers and not decision tree regressors. For the algorithm, the goal is to determine the outcome which minimises the parameters for impurity

$$s = \arg\min_{s} G(D, s), \tag{B.8}$$

while the impurity can be calculated using,

$$G(D, s) = \frac{N_{\text{left}}}{N} H(D_{\text{left}}(s)) + \frac{N_{\text{right}}}{N} H(D_{\text{right}}(s)), \tag{B.9}$$

where $H(D)$ is the impurity function. The input data is in the form of training vectors $x_i \in R^n, i = 1, ..., l$ and the label vector $y \in R^l$. The data is partitioned into two subsets, given by D, at a given node m, with each possible split $s = (i, t_m)$ made up of feature (i) and threshold (t_m). These subsets are calculated using

$$D_{\text{left}}(s) = (x, y)|x_i < t_m$$

and,

$$D_{\text{right}}(s) = D \ D_{\text{left}}(s).$$

The impurity functions, used to calculate G are similar to that used by Spark. The *Gini* impurity function is given by

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$

and the *Cross-Entropy* function is given by

$$H(X_m) = -\sum_k p_{mk} log(p_{mk}).$$

The proportion p used in the impurity functions represents the number of observations (k) $(0, ..., K-1)$ in node m

$$p_{mk} = 1/N_m \sum_{x_i \epsilon R_m} I(y_i = k)$$

with N_m (number of samples for node m). The algorithm is run until maximum depth is reached, $N_m <$ min(samples) or $N_m = 1$.

B.2.3 Naive Bayes algorithm in Spark and Scikit-learn

The discussion as from Scikit-learn (2017) as discussed in Spark, is divided into the multinomial naive bayes algorithms provided and secondly the bernoulli naive bayes algorithm variants. Provided a class variable y and dependent feature vector $x_1, ..., x_n$, the theorem is given by

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}.$$

When applying the independence assumptions (Naivety) the function sums each outcome with the variable probabilities

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_1|y)}{P(x_1, ..., x_n)}, \tag{B.10}$$

with $P(x_1, ..., x_n)$ being a constant given input. When applying the data to multinomial distributed data, the naive bayes algorithm can be adapted. As stated in Scikit-learn (2017), the multinomial naive bayes is typically applied to text classification where the data is represented as word vector counts. As outlined, the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y is given by θ_{yi}

$$\hat{\theta_{yi}} = \frac{N_{yi} + \alpha}{N_y + \alpha n}. \tag{B.11}$$

In (B.11), N_{yi} is the count of feature i in a sample with class y in a given training set T, with N_y the count of all features for class y. If it is determined that the data conforms to a multivariate dataset where each feature is assumed to be binary-valued (1 or 0), then the bernoulli naive bayes algorithm variant can be employed. The algorithm is given by

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i) \tag{B.12}$$

which as stated in Scikit-learn (2017), penalises class y for a feature i not present.

B.2.4 Linear support vector machines algorithm in Spark and Scikit-learn

The SVM algorithm provided by spark is limited to binary classification, as such, the algorithm employed by Scikit-learn (2017) will only be used for binary classification, in order to maintain comparability. The literature regarding the SVM algorithm for the Spark Analytics Program is near identical, therefore to avoid repetition, only the literature from Scikit-learn (2017) description is provided. Provided a vector $x_i \in \mathbb{R}^p$, $i = 1, ..., n$ and label vector $y \in \{1, -1\}^n$ of two classes, the decision function for SVM is,

$$\text{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho) \tag{B.13}$$

where the following ω values are minimised,

$$\min_{\omega,b,\zeta} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n} \zeta_i \tag{B.14}$$

which are subject to

$$y_i(\omega^T\phi(x_i) + b) \geq 1 - \zeta_i$$

and

$$\zeta_i \geq 0, i = 1, ..., n.$$

The decision function divides the respective values into a class. The respective *dual* problem, is used in Scikit-learn to solve for non-linearly separable problems. This is not used in Spark, as it is configured only for linear SVM problems, but is provided for completeness. The *dual* equation is

$$\min_\alpha \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{B.15}$$

subject to

$$y^T\alpha = 0$$

and

$$0 \leq \alpha_i C, i = 1, ..., n,$$

where e is the vector of ones, $C > 0$. The respective kernels $K(x_i, x_j)$ used in the decision function can be linear, polynomial or sigmoid functions. In the function constraints, $Qij \equiv y_i y_j K(x_i, x_j)$ is a $n \times n$ semi-definite matrix and the kernel used is $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)^T$, with vectors mapped into a higher dimensional space ϕ.

B.2.5 Multilayer perceptron classifier algorithm in Spark and Scikit-learn

The multilayer perceptron classifier (MPC) is an ANN and therefore a *supervised* learning technique and is the only ANN available in Spark, therefore used in this project for both the BDAD and standard system. The MPC algorithm will be discussed using the notation from Scikit-learn (2017) to outline the method by which the algorithm classifies instances on Spark and Scikit-learn. As stated in Scikit-learn (2017), the advantages of the MPC are the ability to learn non-linear models, as well as performing the analysis in near real-time.

Provided with a training set $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ and $x_i \in \boldsymbol{R^n}$ and $y_i \in \{0, 1\}$ an MPC conducts the learning phase to predict the outcome using the function

$$f(x) = W_2 g(W_1^T x + b_1) + b_2 \tag{B.16}$$

with W_1 and W_2 weights of the input and hidden layers before final output. A bias factor for the input b_1 and b_2 for the output layers is then added to account for a shift in the sigmoid function, to provide the different nodes in an ANN with a 'constant value' to improve the prediction. The activation function $g()$ used is,

$$g(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \tag{B.17}$$

which simplifies to the logistic function for binary classification and a softmax function. A softmax function calculates the probabilities for each class over all the possible classes (Polamuri, 2017) and is given by

$$\text{softmax}(s)_i = \frac{\exp(s_i)}{\sum_{l=1}^{k} \exp(sl)} \tag{B.18}$$

where the s represents the $i - -th$ element, with K number of classes, as in Scikit-learn (2017). A vector is then produced of probabilities (f(x)) of x belonging to each of the given classes. The output from this function being the class with the highest probability. To measure the difference between the predicted values and the actual values, the MPC uses loss functions. The loss function as in Scikit-learn (2017) is given by

$$\text{Loss}(\hat{y}, y, W) = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y}) + \alpha ||W||_2^2. \tag{B.19}$$

The regularisation parameter is inserted to add any penalties given different input values, to provide the appropriate scaling, with $\alpha > 0$ controlling the penalty magnitude. The weights (W) are initially randomly assigned, and by updating the weights by performing gradient decent, the loss is iteratively decreased. Next, the weights are given by

$$W^{i+1} = W^i - \epsilon \bigtriangledown \text{Loss}_W^i \tag{B.20}$$

which are reduced for each iteration i and previous loss function, with the addition of a learning rate $(\epsilon > 0)$. The stopping condition for this function is the number of iterations set by the analyst or if the improvement in the loss falls below a specific value as stated in Scikit-learn (2017).

B.3 Clustering machine learning techniques

The following is literature provided by Spark (2018) and Scikit-learn (2017), around the method by which the k-means ML algorithm performs clustering.

B.3.1 k-means algorithm in Spark and Scikit-learn

As outlined by Scikit-learn (2017), the k-means algorithm separates the data into groups of a specified size, by determining for each group a collection of data points that will ensure an equal variance value. These groups or number of clusters (k) are predetermined by an analyst, and the variance is calculated by determining the sum-of-square distance each data point is from a cluster centre. Using the notation by Scikit-learn (2017), given a dataset $\boldsymbol{X}$, N number of samples within each dataset and C for the number of clusters that are chosen. The goal of the k-means algorithm is to minimise

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_j - \mu_i||)^2 \tag{B.21}$$

which is the distance each point is from a centroid j, with other μ_i being the mean of the sample values of cluster j. The algorithms by Spark and Scikit-learn use this measure in an iterative manner to determine which sample points belong to a given cluster. The first step in this process is to randomly place initial cluster centroids, with each datapoint being assigned to one of these centroids. The distance is then measured using (B.21) between each data point and centroid (cluster centre) and stored. The second step takes the mean distances of all sample datapoints to the assigned centroids and assigns the new centroid location to the point where these distances are minimised, and then re-calculating this distance between the data points and new centroids. The difference in the old cluster-centroid value (location) and the new cluster-centroid is calculated. This process is repeated until the difference between the old and new cluster centroids reaches a given threshold and the algorithm stops.